Università degli Studi di Milano-Bicocca

Dipartimento di Informatica, Sistemistica e Comunicazione

Corso di Laurea Magistrale in Data Science

Aspect-based Sentiment Analysis su recensioni di film

Relatore: Roberto Boselli

Tesi di Laurea Magistrale di:

Matteo Provasi

Matricola 782922

Anno Accademico 2019-2020

Indice

Capitolo 1

Introduzione

1.1 UGC (User-Generated Content)

Gli **UGC** (User-Generated Content) sono uno degli elementi fondanti ed enfatizzati dal Web 2.0. Dai primi anni del terzo millennio gli utenti hanno iniziato a creare contenuti testuali, multimediali all'interno della rete e con il passare del tempo, grazie alla continua diffusione di internet a livello globale, il volume degli UGC ha continuato a crescere in modo esponenziale. Gli UCG si caratterizzano per tre elementi:

- Sono contenuti generati dagli utenti e non da aziende o dai proprietari che gestiscono un sito.
- Questi contenuti sono creativi e aggiungono informazioni a quello che era già stato creato in precedenza.
- Generalmente questi contenuti sono accessibili anche agli altri utenti.

Esistono diverse tipologie di UGC, ognuna con le proprie caratteristiche, le più comuni sono rappresentate nella *Figura 1.* [1]

Ogni quadrante del grafico comprende un gruppo di UGC con delle caratteristiche in comune, il posizionamento all'interno di essi rappresenta una loro sfumatura: un grado di oggettività sull'asse verticale e sull'asse orizzontale lo scopo principale per cui sono creati. Nel quadrante di intrattenimento sono presenti video, contenuti videoludici o virali, molto comuni all'interno dei Social Media il cui scopo principale è appunto quello dell'intrattenimento, di rendere consapevoli più persone su un

Figura 1.1: Rappresentazione schematica delle principali tipologie di UGC

certo tema e sono caratterizzati da un giudizio soggettivo. Nella parte più oggettiva si hanno i contenuti educativi, articoli, guide o infografiche, esempio di UGC multimediale. Sempre nella parte razionale ma più indirizzata all'acquisto di un oggetto esistono UGC derivanti dalla comparazione di prezzi di diversi siti o delle specifiche di un prodotto. Nella parte più soggettiva il campo è prevalentemente occupato da recensioni e giudizi di utenti molto spesso riguardanti un bene di consumo.

Analizzare i contenuti generati dagli utenti permette di avere una visione più completa della qualità di un prodotto, un servizio o più globalmente capire l'opinione generale rispetto ad un tema. Queste informazioni rappresentano un aiuto nei processi organizzativi e di decision making [2]. UGC come le recensioni sono spesso

altamente considerate da parte degli altri utenti, secondo una ricerca di TripAdvisor [3], il 96% delle persone ritiene essenziale leggere recensioni prima di prenotare un ristorante o un albergo.

Parallelamente al crescente volume di UGC e della loro importanza, aumenta anche il bisogno da parte di enti, aziende, attività commerciali di dover analizzare questi contenuti per comprendere i punti di forza dei propri servizi e su quali ambiti migliorare. Per una persona non è complicato cogliere il senso di un testo creato da un altro utente ma questi contenuti sono di gran lunga superiori a quelli che una persona può manualmente analizzare. I computer permettono di elaborare una quantità di contenuti estremamente superiore, ma hanno il grosso limite di non ragionare come una persona: un computer lavora esclusivamente sul testo in input senza considerazioni esterne. Questa fondamentale limitazione è uno dei motivi per cui secondo *Schuckert et al.* [4] l'analisi degli UGC ha un grande successo a livello di ricerca, ma sul lato pratico e applicativo l'implementazione di architetture per l'analisi di questi contenuti non ha ancora una diffusione così capillare, sia per problemi di scalabilità sia di affidabilità dei risultati.

Per sfruttare al meglio le potenzialità dei computer è necessario modificare i testi dati in input, creare delle regole da cui la macchina può apprendere per avvicinare sempre di più il livello umano; in questo ambito si parla di **NLP** (Natural Language Processing), un campo che comprende elementi di linguistica, computer science ed intelligenza artificiale che si occupa dell'interazione che hanno i computer con il linguaggio naturale. Nel corso degli anni sono state implementate ed affinate diverse tecniche per rendere l'analisi del testo più semplice e ad oggi tutti i principali software hanno delle funzioni per poter gestire in modo appropriato il linguaggio naturale. Le applicazioni principali del NLP sono:

- Traduzioni: traduzione automatica di un testo in un'altra lingua. Questo compito è risultato sempre molto complicato in quanto si deve far corrispondere un testo in una lingua diversa che può avere anche delle strutture sintattiche completamente diverse. Uno dei principali progressi è stato realizzato da Google nel 2016 con l'implementazione di una nuova rete neurale che si aggiorna anche con il contributo degli utenti e dalla valutazione delle traduzioni effettuate [5].

- **NLG**: sigla per Natural Language Generation, le tecniche attraverso le quali viene generato automaticamente del testo di senso compiuto seguendo le regole

semantiche e sintattiche. Le applicazioni principali sono nei chat bot che permettono di comunicare in tempo reale con degli utenti.

- Sentiment Analysis: la classificazione del testo in base al giudizio espresso dagli utenti. Uno dei compiti più scontati ma anche più importanti per identificare quale sia la polarità di un commento, solitamente positiva, negativa o neutra, riguardo ad un prodotto o ad un servizio.

La Sentiment Analysis può fornire delle informazioni molto importanti e non risulta più un compito molto complicato: molte delle tecniche implementate nei software fanno riferimento a dei vocabolari a cui ogni parola è assegnata una polarità e in base al segno del valore complessivo del sentiment di una frase la si classifica di conseguenza. Per la lingua inglese esistono diversi vocabolari con range di valori diversi che si adattano più o meno bene a seconda dell'ambito di applicazione. Nella sua capacità informativa, la Sentiment Analysis ha anche un grande difetto, ovvero quello di raggruppare in un unico output tutta la complessità espressa da una frase. Come evoluzione del metodo è stata proposta una Aspect-Based Sentiment Analysis che mira ad identificare il sentiment dei diversi aspetti trattati in un testo. Il compito si complica in quanto devono essere correttamente individuati i temi trattati da un utente in un suo commento, un lavoro assolutamente non banale per un computer, che però porta a dei risultati più specifici e precisi e di conseguenza anche informativi.

Le analisi proposte in questo lavoro tratteranno UGC relativi a delle recensioni (in lingua inglese) di film su cui verrà sviluppato un workflow per l'implementazione di una Aspect-Based Sentiment Analysis: dai commenti verranno estratte delle tematiche salienti su cui sarà valutata la polarità.

1.2 Letteratura

Con la proliferazione di recensioni, giudizi, raccomandazioni sin dalla nascita del web 2.0 e amplificati con la nascita di piattaforme come i blog, social media e social network, le opinioni online si sono trasformate in quella che può essere considerata come una moneta virtuale per il business con cui le aziende possono consolidare la loro posizione nel settore di mercato e identificare nuove opportunità. In con-

comitanza con l'espansione di questo genere di contenuti, prendeva anche piede la tecnica della Sentiment Analysis, per migliorare le probabilità di tagliarsi una fetta importante del mercato.

Agli albori della Sentiment Analysis, gli studi riguardavano la creazione di vocabolari per definire il sentiment di alcune parti del discorso. Il pioniere di questo lavoro è stato *Hatzivassiloglou* (1997) con l'estrazione del sentiment su degli aggettivi [6], studio poi ripreso qualche anno più tardi anche da *Wiebe* (2000) [7] mentre lo studio di *Riloff, Wiebe, Wilson* (2003) si è concentrato sui nomi [8].

Parallelamente altre pubblicazioni, come quella di Volcani, Fogel (2001) [9], si concentravano sull'analisi delle singole parole all'interno della frase per definire una scala di emozioni e di relazioni con i sinonimi. Questa soluzione non ha trovato molte applicazioni e già negli anni immediatamente successivi si è preferito un approccio più intuitivo seguendo l'idea dei primi studi attraverso i quali si consideravano due singole polarità, positivo e negativo, senza gestire le sfumature di emozioni. *Turney* [10] e *Pang* [11] sono stati i primi ad applicare questo approccio in due paper distinti, il primo su un'analisi di recensioni di prodotti, il secondo su recensioni di film. Questa binarizzazione della polarità, benché sia molto semplicistica, ha trovato e trova ancora oggi diverse applicazioni ed è anche alla base dello sviluppo di metodologie più avanzate. Sempre *Pang* qualche anno più tardi ha sviluppato un modello per la classificazione del sentiment su una scala di giudizio non binaria ma definita da un sistema a 3/4 stelle [12].

Il primo tentivo di effettuare una Aspect-Based Sentiment Analysis è il lavoro svolto da *Snyder* (2008) con l'identificazione di aspetti nelle recensioni di ristoranti [13]. Il punto focale del lavoro era la creazione di un *agreement model* con lo scopo di monitorare il grado di soddisfazione o insoddisfazione all'interno della recensione: se il sentiment si fosse mantenuto costante, allora tutti gli aspetti sarebbero stati classificati con il giudizio generale (sistema basato su un punteggio a stelle); mentre nel caso si fossero riscontrate delle variazioni, l'algoritmo avrebbe cercato di individuare le parti in cui il sentiment si modificava e di collegarle con i relativi aspetti. Le performance in termini di accuratezza erano certamente migliori di quello che viene definito un *random guess* ma significativamente inferiori ai livelli ottenuti in letteratura su un problema non suddiviso in aspetti. Una delle principali problematiche dell'Aspect-Based Sentiment Analysis è quella di identificare correttamente gli aspetti, l'argomento di cui si parla in una parte specifica del testo. Nel corso degli

anni, attraverso la ricerca, si sono costruiti dei vocabolari che permettono di avere un'idea a priori di quello che possa essere un aspetto. Un esempio di questa applicazione è la pubblicazione di *Weismayer, Pezenka, Gan* (2018) nella quale, attraverso il software RapidMiner Studio, si è confrontata la qualità della sentiment su diversi aspetti prima mediante una codifica umana e successivamente con un modello di machine learning [14]. Come confermato dagli stessi autori, l'utilizzo dell'algortimo è limitato al loro ambito, recensioni di ristoranti e hotel, in quanto i diversi aspetti, qualità del cibo, atmosfera, rapporto qualità prezzo ed altri, erano già memorizzati all'interno del modello.

Un approccio più difficile, più ambizioso ma sicuramente più versatile è quello di costruire un algoritmo che permetta di estrarre la valutazione di diversi aspetti non conoscendoli a priori ma adattandosi al contesto dei commenti. Un esempio di ricerca secondo questa mentalità è stato realizzato da *Tun Thura Thet, Jin-Cheon Na, Christopher S.G. Khoo* (2010) sulle recensioni di film [15]. Ad eccezione di alcune particolarità legate al contesto, come il dizionario utilizzato con i punteggi di sentiment e alcuni passaggi della pulizia dei dati, il modello creato è in grado di poter essere esteso anche ad ambiti diversi. La particolarità principale di questo lavoro è l'estrema attenzione con cui i commenti sono stati trattati a livello grammaticale: per identificare gli aspetti le frasi sono state analizzate nella loro complessità, a livello di subordinate, di singole parole e confrontando lo schema delle parti del discorso con quelli che tradizionalmente si trovano nella lingua inglese. Con questa classificazione minuziosa delle frasi risultava più facile l'identificazione degli aspetti in quanto a seconda del tipo di frase in input si valutavano appropriatamente i termini presenti. Un altro esempio di un algoritmo non legato al contesto è la pubblicazione di *Afzaal, Usman, Fong* (2018) nella quale gli autori propongono un nuovo framework con l'accostamento ad ogni parola di un tag identificativo per la parte del discorso e determinate sequenze di tag possono rappresentare porzioni di frasi e quindi facilitare l'identificazione degli aspetti [16].

L'estrazione degli aspetti può essere realizzata anche mediante dei modelli statistici generativi, come nel caso di *Brychcín, Konkol, Steinberger* (2014) [17] mediante l'implementazione di una **LDA** (Latent Dirichlet Allocation) [18] e di *Mukherjee* (2018) tramite un nuovo modello applicato per la prima volta nel suo studio [19]. I modelli generativi permettono di ragruppare parole e gruppi di parole tramite dei collegamenti non direttamente osservabili nel testo, permettendo quindi la creazione di aspetti.

Un altro aspetto da tenere in considerazione per la Sentiment Analysis e più in generale nel NLP è la valutazione di commenti ironici e sarcastici. È abbastanza ricorrente trovate negli UGC e anche nel linguaggio di ogni giorno delle frasi il cui significato è l'opposto di quello letterale. Per le persone non è difficile cogliere queste sfumature, ma per un computer, senza delle apposite procedure, una frase non ha altri elementi oltre alle parole stesse con cui è composta. L'importanza di gestire accuratamente le forme ironiche e sarcastiche è stata sottolineata in diverse pubblicazioni, fra le prime *Reyes et al.* nella quale gli autori discutono di come la gestione di queste particolarità non sia un aspetto da trascurare in quanto il reale da attribuire è l'opposto di quello che è letteralmente presente [21]. Oltre a non essere un compito di facile realizzazione, non esistono grandi ricerce sull'argomento, una delle prime pubblicazioni, *Ghosh et al.* (2015), cercava di valutare l'impatto del linguaggio figuravo su dei *tweet* [20], successivamente sono state proposte delle tecniche, come quella di *Patra et al.* con la *Sentiment abruptness measure*, una metrica che quantifica come e quanto rapidamente varia il sentiment all'interno di un commento [22]. Secondo gli autori più il cambiamento di sentiment è repentino e ristretto in poche parole, allora verosimilmente si tratta di un linguaggio ironico. Gli stessi autori sottolineano come nel loro lavoro, la maggior parte dei commenti ironici era identificata principalmente nei tweet con un sentiment estremamente positivo o negativo.

Capitolo 2

Dati

L'intero progetto, dai dati alle analisi, è stato realizzato tramite il linguaggio Python utilizzando l'interfaccia Jupyter Notebook [23] di Anaconda [24]. I dataset non sono stati reperiti da fonti esterne ma sono stati creati mediante l'utilizzo di uno scraper. Il web scraping è la procedura mediante la quale si possono estrarre informazioni da elementi, strutturati e non, presenti su una pagina web. L'attività si può sintetizzare in due punti:

- Web crawling: riguarda la parte di download e visualizzazione della pagina. In questo primo passaggio si accede alla pagina di interesse e la si visualizza come si farebbe normalmente durante una navigazione sul web.
- Estrazione: le pagine web utilizzano un linguaggio di mark-up per la visualizzazione degli elementi da parte del browser, il più comune è l'**HTML** (Hypertext Markup Language). Visualizzando il codice sorgente HTML è possibile identificare gli elementi di interesse che potranno essere salvati in modo strutturato.

2.1 Questioni legali

Accedere a delle pagine web tramite un bot per estrarre delle informazioni ha fin da subito visto contrapposte le posizioni di chi detiene i dati, e quindi ritiene non

sia lecito ottenere informazioni in 4 questa maniera, e gli utenti che vogliono far uso di queste informazioni [25]. Di base l'utilizzo di web scraper è legale e implementato anche su diversi linguaggi di programmazione, tra cui anche Python, inoltre diverse aziende come Amazon e Google hanno messo a disposizione degli strumenti di scraping. Al contrario non è sempre legale il modo in cui un web scraper viene utilizzato. I siti possono specificare nelle loro condizioni di utilizzo le limitazioni imposte ai web scraper anche se l'eventuale infrazione di esse non è ben definita a livello legale.

Una linea giuda generale sullo scraping chiama in causa l'uso personale delle informazioni ricavate: se un utente estrae delle informazioni ma non le distribuisce a terzi, questo comportamento viene solitamente considerato corretto. La distribuzione di queste informazioni può portare alla creazione di un sito concorrente o alla comparazione di dati provenienti da siti diversi ma riguardanti lo stesso ambito. La distribuzione dei dati ottenuti con lo scraping può quindi intaccare l'attività del sito e di conseguenza una perdita dal punto di vista monetario. A complicare ulteriormente la questione, si aggiunge il fatto che anche l'uso personale comprende delle zone grigie in cui non è sempre ben chiaro quali siano le attività legali.

Un altro aspetto da tenere in considerazione quando si utilizzano dei web scraper è il primo punto della sua attività: il web crawling. Per poter scaricare delle informazioni si deve prima accedere alla pagina e quindi fare una richiesta al server di poterne visualizzare i contenuti. Un bot, se non controllato a dovere, può effettuare un numero di richieste al server notevolmente superiore a quelle di una persona, portando, in casi estremi, alla navigazione rallentata sul sito o addirittura il crash. Esistono dei precedenti di aziende colpite da queste problematiche: nel 2015 la società QVC ha avuto un crash del sito per due interi giorni a seguito di un eccesso di richieste con una frequenza media di 200-300 richieste al minuto con picchi di addirittura 36000 richieste al minuto [26]. L'azienda non ha potuto offrire ai suoi utenti la possibilità di effettuare acquisti online per 48 ore con una conseguente perdita di profitto non indifferente. L'azienda ha presentato la propria causa alla corte distrettuale della Pennsylvania chiedendo un rimborso per l'interruzione del servizio da loro offerto, delle spese processuali e di inibire l'accesso al sito ai web scraper [27]. L'azienda tuttavia non ha saputo portare delle prove decisive a supporto della loro tesi e in combinazione all'assenza di filtri per gestire richieste anomale e con elevata frequenza, all'assenza di un robot.txt (un file in cui un sito può specificare su quali pagine l'accesso dei web scraper è severamente vietato) aggiornato, non è

riuscita a vincere la causa e ad ottenere un risarcimento [28].

2.2 Internet Movie Database

IMDB (The Internet Movie Database) (`https://www.imdb.com/`) è un sito online lanciato nel 1993 che si occupa principalmente di raggruppare informazioni su film, programmi e serie televisive, videogiochi e streaming. Sul sito sono presenti oltre 6 milioni e mezzo di titoli, informazioni sugli attori, valutazioni e recensioni da parte degli utenti. Sul sito è stata effettuata una procedura di scraping per ottenere informazioni sui titoli dei film, gli attori protagonisti e le recensioni da parte degli utenti con la relativa valutazione. Tutte le pagine visitate non rientravano nella lista presente nel robot.txt del sito nella versione di novembre 2019 [29].

2.3 Scraping

La procedura di scraping, così come il resto delle analisi, è stata implementata con il linguaggio di programmazione Python. Selenium è un framework che permette di effettuare test automatizzati con applicazioni web ed è anche il nome dell'omonima libreria utilizzata per stabilire una connessione fra lo script e il browser. Nella sua complessità Selenium è organizzato in quattro elementi, ma per lo scraping è necessario utilizzarne una sola, la parte WebDriver la cui architettura è rappresentata nella Figura 2.1.

Le funzionalità delle componenti sono le seguenti:

- Selenium Language Bindings: è il linguaggio di programmazione che contiene lo script e le azioni che dovranno essere eseguite sul browser. Sono le righe di codice che sul notebook richiamano le funzioni di Selenium per poter aprire una pagina del browser.
- JSON Wire Protocol: gli script sono convertiti in Json API per inviare la richiesta al server e per facilitare lo scambio di dati fra il client ed il server.

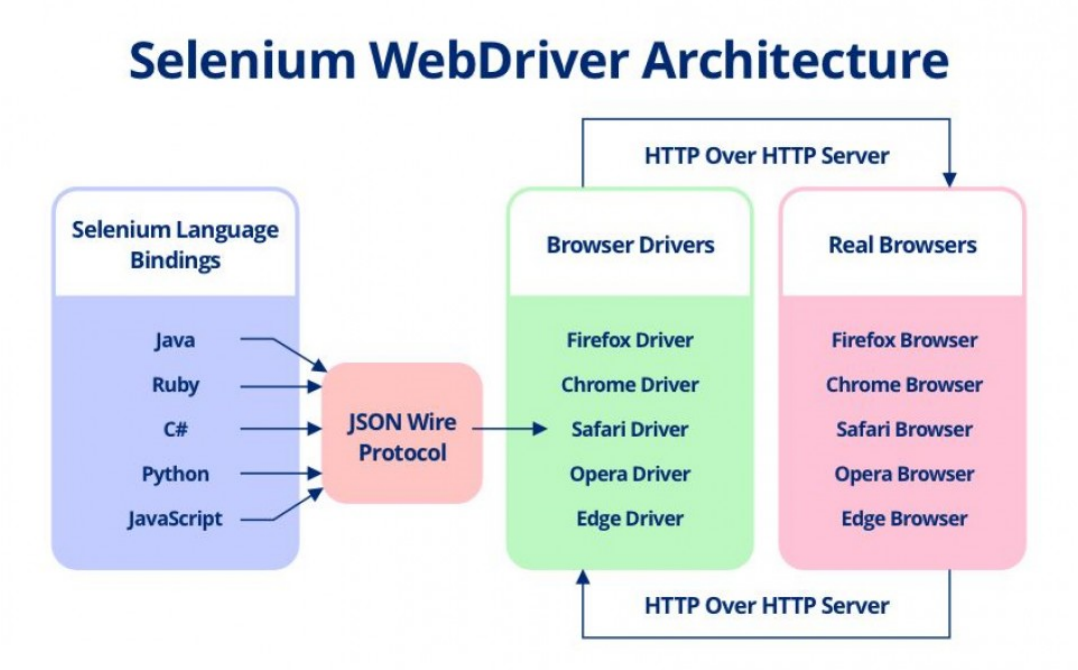

Figura 2.1: Architettura di Selenium WebDriver

- Browser Drivers: questi driver servono per collegarsi al browser vero e proprio in quanto la logica interna e le funzionalità di un browser non sono rese pubbliche dagli sviluppatori e permettono di stabilire una connessione sicura.
- Real Browser: è il luogo in cui le richieste sono inviate e dove verranno eseguite le richieste specificate nello script.

Il browser utilizzato per lo scraping è stato Mozilla Firefox. Fino alla versione 47 di Firefox e della libreria Selenium 2, il procedimento per aprire una finestra del browser dal notebook seguiva esattamente lo schema riportato in precedenza. Nelle versioni più recenti, Selenium non possiede un'implementazione nativa di Firefox e quindi non è possibile stabilire un connessione senza l'ausilio di un file esterno. Sul sito di Firefox è possibile scaricare un file denominato GeckoDriver.exe [30] che permette di stabilire la connessione fra Selenium e Firefox, l'unico vincolo è che la directory dell'eseguibile sia richiamata correttamente dallo script. Per non appesantire la navigazione attraverso il sito e lo scambio di dati, sono state assegnate delle impostazioni al Browser Driver per evitare il caricamento di immagini o file flash.

2.4 Identificazione degli elementi

Una volta visualizzata la pagina di interesse sul browser è stato necessario individuare gli elementi da scaricare. Gli elementi sulla pagina sono rappresentati in modo tale da risultare intuitivi e leggibili all'utente; Selenium offre la possibilità di individuare elementi in base al testo rappresentato, ma questa soluzione non è utile per uno scraping che richiede di identificare più elementi all'interno di una pagina. La via alternativa è quella di passare mediante il linguaggio macchina della pagina web, l'HTML e da questo codice individuare le informazioni da estrarre.

Selenium permette di selezionare gli elementi tramite il relativo Xpath (XML Path Language) presente nel codice HTML. L'Xpath è un linguaggio che permette di identificare uno o più elementi nella pagina web effettuando una sorta di query nel linguaggio markup della pagina stessa. L'Xpath risultante non è altro che una stringa al cui interno si trovano una serie di tag e attributi del linguaggio HTML che definiscono il posizionamento dell'elemento all'interno della pagina web. La ricerca può essere effettuata tramite singoli elementi, specificando un Xpath completo che identifica un elemento univoco all'interno della pagina, altrimenti è possibile effettuare una ricerca per classi di oggetti con la quale verranno visualizzati tutti gli elementi appartenenti ad una specifica categoria. Per individuare un Xpath è necessario utilizzare una delle funzioni built-in del browser con la quale è possibile accedere al percorso di un elemento selezionato.

Tutte le informazioni raccolte nella procedura di scraping mediante le funzioni presentate nelle prossime sezioni, hanno seguito questa logica.

2.5 Funzioni di scraping

Le informazioni estratte possono essere classificate in tre categorie:

- Recensioni: sono le recensioni degli utenti relative ad un film. Ogni film ha la sua pagina dedicata in cui sono elencate tutte le recensioni scritte dagli utenti.
- Titoli: il titolo del film di cui si estraggono le recensioni. Come verrà spiegato successivamente, l'estrazione del titolo è una passaggio fondamentale per la

Sentiment Analysis. Per via di problemi di geolocalizzazione che verranno approfonditi nella relativa sottosezione, è stata creata una funzione apposita per ricavare il titolo di un film.

- Persone: per ogni film sono stati ricavati i nomi dei registi, degli attori e dei personaggi interpretati nei film.

Per ogni categoria è stata creata un'apposita funzione di scraping in quanto le informazioni relative erano distribuite su pagine differenti. Selenium permette di replicare tutte le funzionalità disponibili del browser per la navigazione, quindi è tecnicamente possibile effettuare dei percorsi di ricerca visualizzando una pagina, tornare indietro ed aprirne un'altra senza chiudere la finestra del browser aperta dal notebook. Questa soluzione, oltre ad essere complicata in quanto si dovrebbero visitare almeno cinque pagine per un singolo film, dato che si ritornerebbe indietro su pagine già viste, ha il grosso difetto di rallentare notevolmente la procedura in scraping per via della grande quantità di memoria temporanea immagazzinata dal browser. Da qui l'esigenza di richiamare una finestra del browser per ogni categoria di informazione da reperire.

Per non lavorare un insieme estremamente eterogeneo, i film presi in considerazione sono quelli classificati nella categoria "Drama" da parte del sito, sia perché raccoglie un numero consistente di opere, sia perché è un genere che gode di grande popolarità ed è facile ricavare un elevato numero di recensioni visitando un'unica pagina. Mediante lo strumento di ricerca avanzata offerta dal sito è possibile ottenere una pagina con 250 film di un determinato genere in un determinato anno. Le motivazioni di questa scelta sono le seguenti:

- I risultati possono essere ordinati per popolarità, quindi per numero di utenti che hanno guardato un determinato film (un utente registrato che ha guardato un film non deve necessariamente rilasciare una recensione), è un modo per massimizzare il numero di recensioni da ricavare con il minor numero di pagine visualizzate.

- La ricerca avanzata permette di specificare la tipologia di opera, escludendo serie televisive o simili che sono comunque catalogate come drama.

- Il numero di film elencati di default è 25 ma modificando l'url risultante da questa ricerca è possibile visualizzarne fino a dieci volte in più. Sarebbe stato

possibile visualizzare anche la seconda pagina dei risultati, ricavando quindi i 500 film più popolari in un anno, ma per le limitazioni del browser citate in precedenza riguardo alla visualizzazione di nuove pagine all'interno della stessa finestra e per via del fatto che già gli ultimi elementi della prima pagina non contenevano un eccessivo numero di recensioni, i benefici di queste soluzioni non avrebbero giustificato il tempo computazionale.

I risultati di questa pagina di ricerca sono stati il punto di partenza per tutte le funzioni di scraping.

2.5.1 Recensioni

L'operazione di estrarre recensioni dal sito è risultata molto dispendiosa dal punto di vista computazionale e il codice è stato avviato più volte nel corso di diversi giorni. Per estrarre recensioni da 250 film in un singolo anno il tempo medio di esecuzione è stato di circa quattro ore. Sono stati presi in considerazione i 250 film più popolari dal 1992 fino al 2009. Una volta visualizzata la pagina di ricerca con i 250 film sono state effettuate le seguenti operazioni:

1. Si è ricavato il titolo e il collegamento ipertestuale che rimanda alla pagina principale del film.

2. Le pagine delle recensioni sono identificate da uno specifico tag all'interno dell'url. Modificando l'url di base del film con quel tag è possibile accedere alla pagina in cui sono contenute tutte le recensioni della relativa opera.

3. La pagina contenente le recensioni è stata aperta su una nuova finestra del browser per evitare le problematiche già espresse in precedenza.

4. Sono stati identificati gli Xpath relativi alle recensioni e ad altre informazioni della recensione come la data e il punteggio della recensione.

5. Ognuno di questi elementi è stato salvato in una lista specifica ed una volta estratto il testo da tutte le recensioni, la seconda finestra del browser è stata chiusa e si è continuato per via iterativa con tutti i 250 film.

I film visualizzati nella pagina di ricerca sono posti in maniera strutturata in un elenco numerato. Questa rappresentazione facilita le operazioni di scraping in quanto è sufficiente identificare la parte del codice HTML in cui è presente questo elenco, ricavare l'Xpath di una recensione e procedere attraverso un contatore per ricavare di volta in volta il film desiderato. Di seguito è riportato un esempio di iterazione all'interno dell'Xpath:

```
for i in range(250):
        title = movie.find_element_by_xpath('/html/body/div[3]
        +'/div/div[2]/div[3]/div[1]/div/div[3]/div/div[' + str(i+1) + ']
        +'/div[3]/h3/a')

        title2 = movie.find_element_by_xpath('/html/body/div[3]
        +'/div/div[2]/div[3]/div[1]/div/div[3]/div/div[' + str(i+1) + ']
        +'/div[3]/h3/span[2]')

        title_list.append(title.text + ' ' + title2.text)
```

Il codice si articola nei seguenti passaggi:

- Il ciclo for genera un contatore che itererà tante volte quante quelle specificate, in questo caso 250.
- Attraverso una funzione di Selenium si identifica l'elemento nella pagina corrispondente all'Xpath immesso come parametro, nel caso rappresentato sarà l'Xpath relativo al titolo di un film nell'elenco riportato dalla pagina di ricerca.
- Un secondo titolo viene estratto in quanto IMDB, per gestire i casi di omonimia nei titoli, aggiunge fra parentesi tonde l'anno di realizzazione dell'opera. Questi due elementi, il titolo e l'anno di produzione, sono identificati da due Xpath differenti.
- I due elementi sono concatenati aggiungendo uno spazio in mezzo. La funzione *text* di Selenium permette di visualizzare il nome dell'elemento ricavato esattamente come è visualizzato dall'utente sulla pagina, altrimenti l'elemento salvato è un codice identificativo non facilmente interpretabile. Il risultato è salvato all'interno di una lista.

L'Xpath di base per tutti i film è lo stesso e cambia solamente la parte finale in cui è inserito il contatore. Quando il contatore è pari a uno, l'elemento trovato sarà il titolo del primo film nella lista, quando è due il secondo e così via. Per via di una differenza di indicizzazione fra i linguaggi Python e HTML il contatore deve essere aumentato di uno ad ogni iterazione. La motivazione dietro questa esigenza risiede nella struttura di Python che utilizza i contatori identificando con lo zero il primo elemento, ma il film in posizione zero nella pagina HTML non esiste. Questo contatore oltre ad essere aumentato di un'unità è anche convertito in stringa in quanto su Python non è possibile concatenare elementi di tipo stringa con numeri interi se non sono considerati come caratteri testuali.

La parte di codice riportata è il primo elemento della funzione di scraping delle recensioni, i titoli non sono ricavati tutti assieme, ma una volta ottenuto un elemento si continua nel resto della funzione. Il passaggio successivo è l'identificazione del collegamento ipertestuale che rimanda alla pagina principale del film, anche questo realizzato mediante la ricerca dell'Xpath. Ricavato l'url la stringa viene modificata inserendo un tag che permette di ottenere la pagina del film in cui sono contenute esclusivamente le recensioni ad esso relative. Per non rallentare il processo di estrazione delle informazioni la pagina con le recensioni viene aperta in una seconda finestra del browser.

Di default IMDB mostra la pagina con solamente le prime 25 recensioni elencate per utilità. L'utilità è una metrica del sito definita come il rapporto fra il numero di persone che ha ritenuto quella recensione utile diviso il numero di utenti che non l'ha ritenuta utile. A differenza della pagina di ricerca, in questo caso non è possibile modificare l'url per mostrare più recensioni e l'unica via percorribile è cliccare in fondo alla pagina sulla casella che permette di caricarne altre 25. Sempre mediante l'Xpath è possibile selezionare degli elementi su cui Selenium può simulare il click del puntatore. I film più popolari raggiungevano quasi dieci mila recensioni totali e sia la navigazione che l'identificazione delle recensioni risultava estremamente rallentata se si fossero caricate tutte le recensioni. Il codice è stato dunque limitato a mille recensioni per film, sia con l'intento di rendere il processo di scraping più veloce, sia per non avere un numero limitato di film che avessero la maggior parte delle recensioni totali.

Questa soluzione funziona anche per la rimozione del rumore. Quando si trova una recensione su un sito, non si può avere la certezza assoluta che quella recensione

sia stata scritta effettivamente da una persona che ha visto quel film o più in generale ha usufruito di un determinato servizio. I casi di recensioni false, soprattutto in ambito alberghiero e di ristorazione, per intaccare o aumentare la reputazione di una struttura non sono così rari. A questo proposito esistono dei precedenti, come TripAdvisor, in cui il sito è stato multato per non aver applicato i dovuti controlli alle recensioni creando un danno o vantaggio economio alle attività commerciali coinvolte [31]. Non si può avere la certezza assoluta, ma è plausibile che alcune recensioni false siano state giudicate dagli utenti come non utili e quindi posizionate in fondo all'elenco e non estratte dallo scraping.
Di seguito è riportato il codice utilizzato per visualizzare più recensioni:

```
click_count = 0
    while click_count < 39:
        try:
            load_more = movie2.find_element_by_xpath(
                        xpath='//*[@id="load-more-trigger"]')
            load_more.click()
            click_count = click_count + 1
            time.sleep(2)
        except (NoSuchElementException, ElementNotInteractableException):
            click_count = 39
```

Il codice si può riassumere nei seguenti punti:

- Viene inizializzato un contatore che è messo come condizione di un ciclo while, le seguenti operazioni saranno effettuate fino a quando la condizione risulterà vera.
- Mediante Selenium si identifica l'Xpath della casella che, se cliccata, mostrerà altre recensioni. In questo caso l'Xpath è identificato non tramite un elemento specifico ma ricercando una classe all'interno del codice HTML.
- Si simula un click del puntatore sulla casella e si aggiorna il conteggio del contatore. Nella riga successiva viene fermata l'operazione del codice per un paio di secondi sia per lasciare il tempo alla pagina di caricare i nuovi elementi, sia per non sovraccaricare il sito di operazioni in tempi brevissimi.
- Nel caso in cui la casella non sia identificata correttamente, oppure non si trova l'elemento perché non ci sono altre recensioni da guardare, il contatore

viene assegnato al valore massimo consentito e si passa alla parte successiva della funzione.

Dato che il valore del contatore è aggiornato ad ogni click per visionare più recensioni, il valore massimo è raggiunto quando sono presenti mille recensioni nella pagina. Il salvataggio del testo di una recensione segue la logica utilizzata per estrarre il titolo di un film: un contatore è stato iterato diverse volte in proporzione al numero di click effettuati e per ogni elemento il testo della recensione è stato estratto e salvato in una lista. Per ottenere i testi delle recensioni si sono dovute specificare diverse casistiche in quanto le recensioni potevano differire di tipologia e di conseguenza anche nell'Xpath:

- Recensioni standard: sono quelle recensioni il cui testo era visualizzato nella sua interezza in modo automatico.
- Spoiler: quando un utente scrive una recensione può contrassegnarla come spoiler se contiene commenti particolari sullo svolgersi della trama.
- Recensioni troncate: sono recensioni come quelle standard ma caratterizzate da un elevato numero di parole; di default per non appesantire troppo la pagina, sono troncate in automatico dal sito.

Il testo delle recensioni standard era estraibile senza problemi, le altre due tipologie avevano una freccia rivolta verso il basso che, se cliccata, avrebbe mostrato il testo nella sua interezza. Come fatto per i click sulla casella dei commenti, è stato implementato un blocco di codice con lo scopo di identificare queste frecce e cliccarle nei casi opportuni. La struttura dell'Xpath di questi elementi variava nel caso in cui si trattasse di una recensione troncata o spoiler.

Una volta identificata la recensione, il testo è stato salvato in una lista, arrivati a fine pagina tutta la lista è stata aggiunta ad un'altra lista più grande in modo da far corrispondere ad ogni elemento di questa lista l'insieme delle recensioni di un singolo film.

2.5.2 Titoli

I titoli dei film non dovranno essere analizzati direttamente tramite la Sentiment Analysis, ma le informazioni contenute in esse sono indispensabili ai fini del proget-

to. I titoli avranno una funzione di filtro, saranno delle parole che non dovranno essere analizzate durante la valutazione del sentiment. Se il titolo del film comprendesse parole con un sentiment non nullo e il titolo venisse citato dall'utente nella recensione, senza un filtraggio si andrebbe a considerare un sentiment diverso da quello che è realmente espresso in quanto il recensore avrebbe solamente richiamato il nome dell'opera. Per esempio nel film *A Beautiful Mind* la parola *Beautiful* ha un sentiment molto positivo ma nei casi in cui si riconoscano all'intorno di essa le parole che compongono il titolo del film, il sentiment dovrà essere considerato neutro.

L'identificazione dei titoli dei film è stata realizzata nel primo passaggio della funzione di scraping, come già spiegato nella sottosezione precedente. Il problema di questo approccio risiede nel modo in cui IMDB gestisce i nomi dei film in quanto il sito applica una geolocalizzazione automatica mostrando dunque i titoli tradotti, in questo caso in italiano, dalla lingua originale. Non avendo a disposizione i titoli originali non è possibile effettuare i dovuti filtraggi. La geolocalizzazione è automatica e può essere cambiata solamente se si registra un account al sito e si effettua il login, tuttavia la pagina di login è una di quelle esplicitamente segnalate nel robot.txt, quindi non era possibile forzare il browser ad effettuare un login.

Per ricavare i titoli originali è necessario passare dalla pagina con la lista dei film più popolari alla pagina principale di un film. In questa pagina è presente sia il titolo tradotto, sia il titolo originale. La funzione per ricavare i titoli originali è stata realizzata separatamente a quella delle recensioni in quanto tutte le operazioni assieme avrebbero comportato l'utilizzo di diverse finestre del browser e quindi un rallentamento delle operazioni. Dalla pagina principale, il cui url era già stato ricavato in precedenza per accedere alla pagina delle recensioni, sono stati individuati gli Xpath relativi al titolo originale e al titolo tradotto che sarà utilizzato come chiave per un successivo matching.

I titoli originali sono stati individuati tramite la ricerca di parole specifiche e non con l'Xpath esatto dell'elemento. I titoli originali erano accompagnati dalla stringa *(original title)*, trovato questo elemento testuale si è ricavato il titolo relativo. Si è preferita questa scelta in quanto i titoli che non hanno traduzione in italiano presentano ovviamente solo il titolo originale e quindi la ricerca per Xpath specifico non era ottimale in questi casi. Da questa funzione di scraping si sono ottenute in output due liste, una con i titoli originali, una con quelli tradotti; nei casi in cui non esistesse un titolo tradotto in corrispondenza di quel film si è inserito un elemento vuoto per mantenere la lunghezza delle liste.

2.5.3 Nomi di attori

La terza ed ultima funzione di scraping è stata implementata per ricavare informazioni riguardanti le persone coinvolte nella realizzazione del film: i registi, gli attori, gli scrittori e i nomi dei personaggi interpretati. Queste informazioni sono contenute in maniera sintetica nella pagina principale di un film in cui sono mostrati fino ad un massimo di tre persone per categoria. Le informazioni complete sono elencate in una pagina a parte, come accadeva per le recensioni.

Analogamente a quanto realizzato per le recensioni, trovato il titolo di un film si modifica l'url in modo tale da ricavare quello della pagina principale e poi si aggiungono dei tag per creare il collegamento ipertestuale che rimanda alla pagina con l'elenco di tutte le persone che hanno preso parte alla realizzazione dell'opera. Le informazioni sono elencate in tabelle per ognuna categoria, dunque esiste una tabella contenente i nomi di tutti gli attori, una tabella con i nomi di tutti gli scrittori e per tutte le altre sezioni. Per ogni tabella è stata creata una sottofunzione specifica con il compito di iterare all'interno dei suoi elementi e di cambiare l'Xpath combinato con un contatore per ottenere le informazioni sulle persone coinvolte. Per ogni categoria è stata creata una lista specifica in cui sono stati salvati i risultati.

I nomi delle persone che hanno preso parte alla realizzazione del film hanno una doppia funzione: come per i titoli alcuni cognomi possono essere delle parole in lingua inglese cariche di sentiment ma che anche in questo caso dovranno essere filtrate; inoltre questi nomi possono aiutare ad identificare l'aspetto di cui si sta parlando in una parte della recensione: se per esempio si citano più attori di fila, è probabile che l'utente stia analizzando la parte recitativa.

2.5.4 Matching dei dati

I dati acquisiti in sessioni differenti sono stati uniti per creare dei dataset specifici: uno per le recensioni con data, titolo del film, valutazione, uno con i nomi degli attori e delle personalità legate al film e un altro con la corrispondenza fra titolo originale e titolo tradotto. La procedura di matching ha consentito di unire le informazioni e di scartare o sistemare gli eventuali errori dovuti allo scraping.

Per le recensioni il primo problema è stato gestire il titolo del film associato poiché le parentesi tonde erano state prese come delimitatore in quando le uniche

occorrenze sarebbero dovute essere, nei titoli tradotti, l'anno di realizzazione, tuttavia in due casi le parentesi erano presenti anche nel titolo di base del film. A seconda del numero di parentesi trovate il titolo è stato separato e riunito opportunamente. Nella funzione di scraping sono state implementate delle righe di codice con il compito di gestire le eccezioni, principalmente i casi in cui un determinato Xpath non era individuato; invece di bloccare l'intera procedura la funzione passava all'iterazione o al passaggio successivo. Sempre nei titoli si è riscontrato un caso anomalo in cui un elemento risultava duplicato. Facendo una comparazione con le corrispondenti liste di recensioni, si è notato che ad un titolo non erano collegate recensioni, mentre all'altro duplicato il processo di scraping aveva funzionato correttamente. L'ipotesi più probabile è che il titolo sia il residuo e, non portando con sé informazioni importanti o non reperibili nell'altro duplicato, è stato rimosso.

Per le date e le valutazioni si sono riscontrati due problemi della medesima entità: la lunghezza della lista contenente le date era di una singola unità più corta rispetto a quella delle valutazioni. La funzione di scraping era stata realizzata in modo tale per cui tutte le informazioni sarebbero state raccolte nel caso fosse stata individuata la recensione e niente, poiché non esistenti, altrimenti. Attraverso un'analisi degli indici delle liste è stato possibile trovare i punti in cui l'algoritmo non ha salvato la data di pubblicazione della recensione. Le pagine delle relative recensioni sono state visitate manualmente ed effettivamente in quei casi la struttura della data e dello score non erano concordi con quello che era il formato tradizionale: fra lo score e la data è presente una tabulazione per separare meglio visivamente gli elementi, ma in due singole occasioni la tabulazione era mancante. Il risultato era la data separata dalla valutazione da un singolo spazio e questa piccola variazione si rifletteva anche nel codice HTML e quindi nell'impossibilità di ottenere l'Xpath corretto. Poiché è stato possibile replicare queste eccezioni, l'errore è da imputare al sito piuttosto che un'errata visualizzazione della pagina durante lo scraping.

Con lo scraping dei titoli il processo non ha avuto particolari problemi, mentre per l'estrazione dei nomi delle personalità sono state necessarie delle modifiche. Per due anni distinti le liste in output non avevano la lunghezza predefinita che dovrebbe essere di 250, un elemento per ogni film al cui interno si trova un'altra lista con tutti i nomi trovati. Le eccezioni hanno coinvolto sia elementi all'inizio che alla fine della lista, e si trattavano di elementi duplicati. Dopo aver effettuato i dovuti controlli, sono stati eliminati gli elementi che erano stati acquisiti in maniera parziale.

Dopo aver ripulito i dati dagli errori di scraping, le informazioni sono state

organizzate in tre dataset, concatenando per riga le informazioni delle diverse liste:

- Recensioni: contiene la data, la valutazione, il titolo tradotto e il testo della recensione.
- Titoli: in una colonna sono presenti i titoli originali e nell'altra i corrispettivi titoli tradotti.
- Personalità: oltre al titolo del film si hanno diverse colonne per gli attori, i registi, gli scrittori e i nomi dei personaggi interpretati.

Nel successivo capitolo saranno presentate le tecniche di pulizia dei dati per rendere le informazioni più accessibili alle tecniche che saranno alla base della Sentiment Analysis.

Capitolo 3

Pre-processing

Analizzare dati testuali da parte di un computer non è un compito facile e per renderlo possibile devono essere applicate delle trasformazioni al testo. Le lingue sono complesse, esistono tante regole grammaticali per generare un testo corretto; per noi è relativamente facile, una volta padroneggiata una lingua, capire se un testo è scritto a dovere o sono presenti errori di sintassi o semantici, per i computer il compito è più arduo. A complicare ancora di più la questione è la possibilità di esprimere un concetto in svariati modi, mediante una costruzione diversa di una frase, con l'uso di sinonimi o in alcuni casi anche del sarcasmo. Per un computer tutti questi elementi sono completamente diversi fra di loro, possono anche rappresentare lo stesso concetto. Da qui nasce la necessità di standardizzare il testo per facilitare i computer nel loro compito di analisi. I risultati saranno paradossalmente dei testi che violano tantissime regole grammaticali e quasi di difficile interpretazione per noi, ma è l'unico modo per aumentare le performance da parte degli algoritmi.

La realizzazione di questo compito è reso ancora più arduo dalla natura stessa dei UCG poiché non sempre le regole della lingua sono rispettate da parte degli utenti. Per esempio una regola di base è quella di iniziare con la lettera maiuscola una frase o di spaziare i segni di punteggiatura e le parole seguenti; molti utenti sono attenti a queste regole, altri meno. A seconda del Social Media di riferimento, gli errori possono essere più o meno frequenti: per un tweet ad esempio il livello di controllo da parte di un utente è esiguo, in molti non rileggono neanche il testo scritto che potrà facilmente contenere anche errori ortografici o di battitura; per le recensioni il contesto sicuramente aiuta dato che per un'analisi critica l'utente deve ragionare su come esprimere le proprie opinioni e verosimilmente il testo sarà

riletto almeno una volta. La lingua inglese è ottimale da analizzare in quanto ha una grammatica relativamente semplice come per la gestione delle forme singolari/plurali o le declinazioni verbali; inoltre è stata l'oggetto principale della ricerca in ambito di NLP e di conseguenza esistono molti più strumenti di analisi a disposizione rispetto ad altre lingue.

3.1 Analisi delle parole

Il modo più semplice di standardizzare i termini è quello di lavorare sulle singole parole piuttosto che su un'intera frase. Identificare i termini è molto semplice in quanto tutte le parole sono divise da uno spazio, prendendo questo elemento come separatore si può accedere ad una singola parola all'interno della recensione. Non esiste uno schema preciso su quale siano le funzioni da applicare per il preprocessing e in che ordine, ma è una scelta dettata dal contesto. La prima parte della standardizzazione applica le seguenti funzioni:

- Rimozione spazi multipli: come già accennato lo spazio è l'elemento fondamentale per identificare i termini e nel linguaggio corretto esiste un unico spazio fra una parola ed un'altra. Non è però raro trovare degli utenti che per errore applicano un doppio spazio bianco per separare due elementi. Dato che Python identifica gli spazi bianchi singolarmente, averne due o più di fila andrebbe a creare degli elementi nulli in quanto sarebbero considerati come singoli delimitatori e non uno solo. Per evitare la creazione di elementi senza contenuto, nel caso si individuino più spazi bianchi consecutivi, essi saranno condensati in uno solo.

- Rimozione numeri: i numeri non sono delle informazioni utili per il sentiment di una frase e in diverse altre applicazioni rappresentano semplicemente un'informazione superflua. Si rimuovono sia i numeri isolati sia i numeri dalle parole alfanumeriche che potrebbero essere state create da errori di battitura.

- Rimozione dei tag: non sono da confondere con gli hashtag popolari soprattutto su Twitter, i tag a cui si fa riferimento sono quelli del linguaggio HTML. Poiché i testi provengono da uno scraper e come detto nei capitoli precedenti

il testo visualizzato non corrisponde sempre a quello che sarà estratto, è possibile che siano presenti dei tag HTML all'interno delle recensioni. I tag HTML sono sempre in coppia e servono per strutturare il testo mostrato a schermo, inoltre esistono dei tag di protocollo che identificano altre particolarità, una fra tutte è la nuova linea rappresentata da */n*.

Queste funzioni sono state applicate ad ogni riga del dataset delle recensioni in cui ogni riga corrisponde ad un'unica recensione. In un solo passaggio dalle recensioni si estraggono le singole parole, per ognuna di esse sono applicate le funzioni di pre-processing elencate in precedenza e al termine di tutta la frase, mediante una *list comprehension*, costrutto particolare di Python, viene ricreata l'intera frase con i termini modificati. Durante questo processo il software ha indicato dei problemi su delle recensioni. In alcuni casi lo scraper non ha identificato bene l'Xpath della recensione e ha salvato un elemento vuoto che stranamente non è stato salvato come stringa, la tipologia di default su Python per queste situazioni, ma come un elemento numerico nullo. L'applicazione delle funzioni di pre-processing su questo tipo di elemento provocava degli errori, risolti convertendo gli elementi in stringhe vuote.

Una delle principali funzioni di pre-processing, che è la rimozione dei segni di punteggiatura, non è stata applicata in questo passaggio per un motivo ben preciso. La rimozione della punteggiatura è essenziale per ottenere dei termini puliti in quanto anche un singolo punto attaccato alla fine della parola rende l'elemento diverso dalla parola in sé. La ragione per cui l'utilizzo di questa funzione è stata posticipata deriva dal fatto che i segni di punteggiatura considerati non sono solo i punti e le virgole, ma anche gli apostrofi e gli accenti. Nell'inglese scritto si ha un uso massiccio delle forme contratte che prevedono la presenza di apostrofi per esempio la forma *I'm* è la contrazione di *I am*, ma la rimozione della punteggiatura porterebbe ad avere un nuovo elemento, *Im*, che non ha alcun significato e sarebbe molto complicato risalire in un secondo momento alla forma originale. Altri casi particolari riguardano contrazioni come *it's* $\rightarrow$ *its* in cui si passa da pronome e verbo a pronome riflessivo che però non avrebbe senso in quella posizione della frase.

Finita la prima parte di pre-processing si è proceduto con la gestione di queste forme contratte: è stato creato un dizionario con chiave il termine contratto e come valore la forma estesa. Ogni volta che nel testo compariva una forma contratta, si ricercava il relativo valore nel dizionario da mettere come sostituzione. Dato

che la rimozione della punteggiatura non era stata ancora effettuara, i termini nel dizionario comprendevano anche alcuni punti e virgole per replicare il testo che si sarebbe effettivamente ritrovato all'interno delle recensioni. Altra particolarità da notare è che in alcune recensioni l'apostrofo era automaticamente sostituito con un backslash e dunque anche queste eccezioni sono state considerate all'interno del dizionario.

3.2 Analisi delle frasi

Per la Sentiment Analysis non è sufficiente lavorare con le parole ma è necessario considerare anche le frasi nella loro interezza. In questa procedura l'attenzione è rivolta alle frasi che saranno delimitate da tre elementi: il punto, il punto di domanda ed il punto esclamativo. Alcuni paper riportavano come elementi solamente i primi due, ma non è raro trovare anche dei punti esclamativi negli UCG. Questo step è un ulteriore motivo per il quale la rimozione della punteggiatura non è stata ancora effettuata, altrimenti si sarebbe complicato l'identificazione delle frasi.

La necessità di analizzare le frasi non è così intuitivo come per le parole in quanto le frasi stesse sono composte da parole e potrebbe essere una ridondanza applicare un pre-processing anche su di esse. Questo step prevede l'identificazione delle frasi all'interno di una recensione e l'eliminazione dei duplicati. Se una recensione è composta da diverse frasi e due di esse sono uguali fra di loro, solamente la prima sarà mantenuta. La ragione per cui si eliminano i duplicati non è per ragioni di spazio o perché non è accettabile avere due frasi identiche ma è per evitare un *sentiment boosting*. Si consideri la frase *This movie is wonderful*, il sentiment sarà elevato in quanto *wonderful* è un termine fortemente positivo. Alcuni utenti possono riproporre una frase, come quella in esempio, per enfatizzare il loro apprezzamento verso il film e per attirare l'attenzione degli altri utenti che leggeranno la sua recensione. Una frase identica ad un'altra, già esistente nella recensione, non costituisce una nuova informazione e il sentiment globale della recensione o dell'aspetto tenderà ad aumentare, o a diminure, a seconda di come sia posta la frase.

Per semplificare il codice, i punti di domanda e i punti interrogativi sono sostituiti con un punto normale, che sarà l'unico elemento discriminante per determinare

la fine di una frase. Per ogni recensione si ottiene una lista di frasi e da questa si scarteranno i duplicati e le frasi con una lunghezza eccessivamente ridotta che potrebbero essere state generate durante la divisione in presenza di più punti esclamativi consecutivi.

3.3 Spelling correction

Una tecnica, che non ha avuto un'ampia applicazione nei lavori di Aspect-based Sentiment Analysis, è la spelling correction. Questa tecnica cerca di modificare le parole che sono state scritte in modo errato per errori di battitura. Un'applicazione banale del metodo è presente nei principali motori di ricerca sul web nei quali, immesse delle parole come query, compare un suggerimento per correggere le parole immesse nella ricerca. I motori di ricerca si basano su diversi fattori per identificare le parole scritte erroneamente e proporre un'alternativa, ad esempio il numero di volte in cui parole o coppie di parole sono comparse assieme nelle query precedenti, dalla visibilità dei risultati ed altro ancora. Negli UGC è difficile trovare delle regole così marcate e quindi la spelling correction non è sempre facile da realizzare e non è sicuro che si ottenga un miglioramento significativo nella correzione dei termini.

Uno dei primi lavori è stato realizzato da Peter Norvig (2009) [32] nel quale, dato un dizionario di partenza, sono realizzate quattro possibili distinte operazioni per modificare i candidati e farli corrispondere ad una parola del dizionario:

- Rimozione: viene rimossa una lettera dalla parola candidata.
- Trasposizione: l'ordine di due lettere viene invertito.
- Sostituzione: una lettera è cambiata con una nuova.
- Inserimento: viene aggiunta una lettera mancante.

Il numero totale di operazioni effettuate è conteggiata come distanza di edit, nello specifico il metodo utilizza la vera distanza di Damerau–Levenshtein [33] [34] in cui ognuna delle operazioni conta come un'unità all'interno della distanza totale ed è possibile modificare delle sottostringhe. Un esempio di questa distanza è il

seguente: *ed("CA" → "ABC")* = 2 dove la prima operazione è la rimozione della lettera centrale e la seconda è la trasposizione delle lettere rimanenti. Questo metodo dovrebbe garantire un'accuratezza intorno al 70% ma ha lo svantaggio di essere estremamente lento al crescere della distanza di edit in quanto ha una complessità cubica.

Diverse altre tecniche sono state implementate per migliorare i tempi computazionali che sono il principale problema del metodo, l'algoritmo implementato è il **SymSpell** (Symmetric Delete Spelling) ideato da Wolf Garbe (2012) [35]. L'algoritmo utilizza solamente la rimozione come operazione di base: per ogni termine nel dizionario di partenza sono effettuate delle rimozioni creando delle nuove parole che saranno salvate all'interno di esso. Si procede con il controllo delle parole candidate effettuando anche in questo caso delle rimozioni e si controllano gli eventuali match all'interno del dizionario. Con un'unica operazione sono gestite tutte le casistiche precedenti, anche gli inserimenti in quanto i termini originali nel dizionario sono modificati con delle rimozioni e questi nuovi termini possono combaciare con le parole da correggere. Dovendo fare un'unica operazione il numero di termini da generare cala drasticamente e di conseguenza anche i tempi di ricerca come evidenziato da dei testi eseguiti dall'autore del metodo e riportati in *Figura 3.1*.

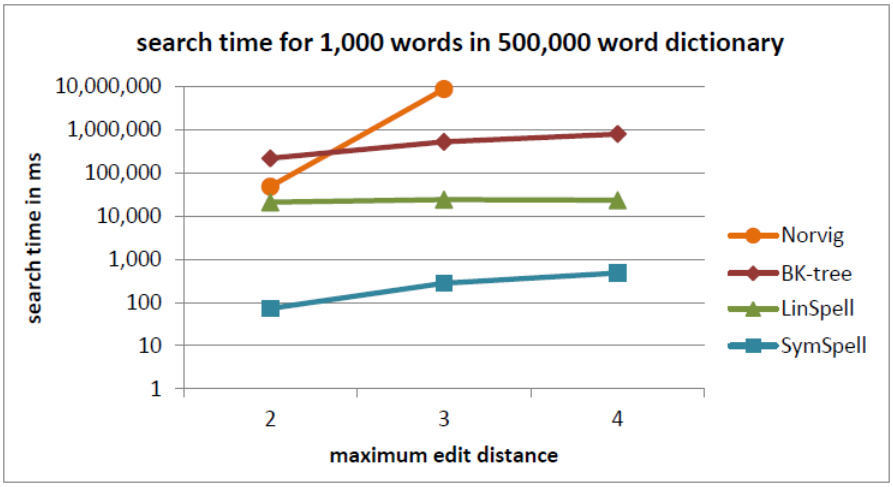

Figura 3.1: Confronto tecniche di spelling correction per tempi computazionali e distanza di edit

Con una distanza di edit pari a tre, l'algoritmo è un milione di volte più veloce di quello di Norvig e sensibilmente migliore di tutti gli altri metodi per qualsiasi distanza di edit.

Il primo step è la selezione di un dizionario di riferimenti su cui saranno applicate delle rimozioni per gestire più errori di spelling possibili. Il dizionario utilizzato comprende circa 300′000 termini in lingua inglese in cui sono state scartate le parole alfanumeriche in quanto sono già state filtrate nella fase di pre-processing. Parallelamente a questa lista si è generato un secondo elenco contenente tutti i nomi delle personalità legate ai film. Non è raro trovare dei nomi e soprattutto dei cognomi che sono molto simili ad un termine realmente esistente nella lingua inglese e che potrebbero essere identificati erroneamente come parole da correggere. I nomi delle personalità sono molto importanti in quanto se citati in una recensione possono dare importanti informazioni sull'argomento della frase. Questa nuova lista funziona come una specie di filtro in cui i termini presenti non dovranno essere controllati.

Nella seguente sottosezione è presentato un altro passaggio del pre-processing rivolto all'identificazione di altri termini che non saranno considerati dall'algoritmo di spelling.

3.3.1 Named Entity Recognition

Per avere una spelling correction più precisa è necessario ampliare ulteriormente il dizionario già creato attraverso una **NER** (Named Entity Recognition). Con questa tecnica si individuano quei termini riferiti a persone, organizzazioni, brand o luoghi che non compaiono in nessun vocabolario della lingua inglese ma non sono di certo dei termini scritti erroneamente. L'Università di Stanford ha messo a disposizione un modello già addestrato per l'identificazione delle entità [36]. Il difetto più grande di questo modello è la lentezza computazionale con il processo che viaggia in media ad un unico termine analizzato al secondo. Fortunatamente l'Università stessa permette di effettuare una connessione sul loro server in modo da velocizzare fino a trenta volte le analisi. Il collegamento è stato realizzato tramite un codice Java nel prompt dei comandi, su Python è poi stata richiamata la funzione di NER specificando la porta di accesso al server creata sul prompt. Gli output possibili del modello sono:

- Termine originale: la parola analizzata non è riconosciuta da modello, si potrebbe quindi trattare di un errore di battitura, una parola straniera, o una sigla non presente nel dizionario.
- Persone: nomi di persone.

- Organizzazioni: associazioni, fondazioni, marchi.
- Luoghi: parole che rappresentano un'entità territoriale spaziando da paesi fino ad arrivare a quartieri di città nel caso degli Stati Uniti d'America.

L'applicazione della NER è un altro motivo per cui in precedenza è stato saltato un passaggio molto comune nelle operazioni di pre-processing, ovvero quello di riportare tutti i termini in minuscolo. Portare tutti i termini in minuscolo crea sia dei problemi nell'identificazione dei nomi degli attori, sia per la NER in quanto i modelli addestrati tengono in considerazione se la parola analizzata inizia con la maiuscola o è tutta scritta in maiuscolo come nei casi delle sigle delle associazioni. L'algoritmo ha analizzato i termini a livello di frase completa poiché il posizionamento di un termine e le parole adiacenti determinano un diverso output del modello rispetto all'analisi di un termine preso singolarmente. Un esempio è l'entità *New York* composta da due parole che sarebbe classificata in maniera errata se si fossero considerati i due termini separatamente.

Il risultato della classificazione è stato talvota ambiguo con alcune parole etichettate come due entità differenti. Di questi l'unico caso realmente problematico era quello in cui una parola è stata catalogata come originale in alcuni casi ed entità in altri in quanto l'obiettivo non era identificare l'entità specifica ma limitarsi ad individuarle. Nel primo scenario la parola potrebbe essere realmente un'entità ma si potrebbe trattare anche di un errore di battitura o una parola straniera. Per queste situazioni al limite è stato deciso di etichettare la parola con l'output più comune offerto dal modello. Tutte le parole sono state inserite in una lista ed ignorate durante la spelling correction.

3.4 POS Tagging

Dall'algoritmo di spelling correction si è ricavato un dataset identico per struttura a quello originale ma con le recensioni ottenute in output dall'algoritmo. Per l'algoritmo si è deciso di prendere come parametro una distanza di edit pari a 2 per i seguenti motivi:

- La lunghezza media ponderata dei termini in inglese (considerando anche la frequenza con cui appaiono nella lingua) è inferiore a cinque, dunque utilizzare un valore troppo alto potrebbe coprire troppi casi, anche molto distanti dalla parola obiettivo.

- Per un motivo correlato si vuole ottenere un buon rapporto fra numero di possibili candidati in output e numero di buoni candidati; in letteratura è quindi consigliato utilizzare questo valore di distanza di edit o al massimo uno superiore.

- Nel lavoro di Norvig il 76% delle parole si poteva correggere con distanza di edit pari a uno e con due la percentuale saliva a quasi il 99% [32].

Non è possibile quantificare esattamente quale sia stata la percentuale di accuratezza raggiunta dall'algoritmo ed entrambi i dataset saranno portati avanti in parallelo con le analisi.

Il passaggio successivo è quello del **POS Tagging** (Part of Speech Tagging), ovvero assegnare ad ogni parola un'etichetta corrispondente alla parte del discorso che ricopre nella frase. Il modello utilizzato è stato preso dalla libreria di Stanford e anche in questo caso è stata utilizzata una connessione al loro server per velocizzare le operazioni. L'Università permette di scaricare più modelli e sarà utilizzato quello consigliato per il rapporto velocità/precisione. Il modello scelto ha un'accuratezza su un corpus base preso come test set del 96.97% mentre il modello migliore in assoluto raggiunge il 97.32% ma è tre volte più lento del primo [37]. Il modello in output restituisce due tipologie di tag:

- **UPOS** (Universal POS tags): è un insieme di 17 tag ognuna delle quali, con qualche eccezione, corrisponde ad una parte del discorso.

- **XPOS** (Treebank-specific POS): è un insieme più specifico che contiene il doppio di tag in quanto si specificano talvolta delle differenze all'interno della stessa parte del discorso, ad esempio i nomi sono divisi fra singolari, plurali.

Etichettare le parole è un passaggio importante per l'Aspect-Based Sentiment Analysis perché come verrà illustrato più avanti, per identificare gli aspetti può essere utile identificare delle sequenze di tag. L'output del modello utilizzato sarà UPOS in quanto non si necessita di conoscere il particolare tipo di tag, ma è sufficiente trovare la parte del discorso. L'algoritmo ha analizzato una frase alla volta e ha

restituito in output una sequenza di tag che sono state salvate in una lista a parte; il procedimento è stato realizzato sia per il dataset originale sia per quello ottenuto dopo la spelling correction. I tag si trovano nello stesso formato delle parole, quindi per ogni recensione esiste una specifica lista con tanti elementi, sequenze di tag, quante sono le frasi identificate all'interno della recensione.

Nel prossimo capitolo saranno illustrate le tecniche per l'identificazione degli aspetti all'interno delle recensioni.

Capitolo 4

Identificazione degli Aspetti

Il progetto punta ad identificare gli aspetti in maniera totalmente automatica per rendere tutto il progetto indipendente dall'ambito e renderlo applicabile anche su testi di altri domini. Il primo passo sarà quello di analizzare i pattern estratti dai tag ottenuti nella sezione precedente e successivamente implementare delle tecniche per il raggruppamento.

4.1 Tagging

Prima di analizzare i pattern si devono fare delle considerazioni sull'algoritmo di tagging realizzato precedentemente. Per come è stato costruito il tagger si dovrebbe avere una corrispondenza perfetta fra il numero di elementi in una lista di parole con il numero di tag identificati. Anche per le parole composte da più termini o le entità come *New York* il tagger, dato che analizzava la frase nella sua totalità e non parola per parola, le etichettava correttamente ma spezzava l'output per mantenere il numero corretto di elementi. Le parole non sono state unite direttamente ai tag sia perché sarebbe risultato più complicato accedere tante volte alle singole parole dovendo applicare delle funzioni per staccarle dal tag, sia per problemi di spazio e di velocità computazionale.

La corrispondenza fra parole e tag è fondamentale in quanto tante delle analisi che sono effettuate da una parte devono corrispondere a delle modifiche anche negli

altri elementi. La mancanza di questa connessione eliminerebbe gli accoppiamenti parola-tag e di conseguenza tutte le successive analisi perderebbero di significato.

Controllando la lunghezza delle liste si è però notato che in diversi casi questa corrispondenza non era verificata e che in tutte le circostanze in qui questo assunto cadeva era per la presenza di una lista di tag più lunga di quelle delle parole. Il problema era quindi da ricercarsi nel tagger e a questo scopo sono stati salvati i termini in tutte le recensioni con problemi di lunghezza e si è effettuato un conteggio. Si è notato che molte delle parole sono derivanti dal fenomeno denominato *relaxed pronunciation*, ovvero quando due sillabe di suono molto simile fra loro sono unite in un'unica. Questa particolarità evidente nella lingua parlata si rispecchia anche nei testi in cui queste particolarità sono ormai considerate delle forme corrette. Ad esempio le parole *"don't know"* nella forma contratta sono riportate come *"dunno"*.

Il modello utilizzato ha implementato un sistema per riconoscere queste forme contratte e gestire i tag di conseguenza: infatti in alcuni casi, come quello presentato, è difficile assegnare uno dei 17 tag alla singola parola dato che sono presenti sia una negazione che un verbo. L'algoritmo risolve il problema in automatico spezzando l'output del tagging e da questa situazione ne risulta una lista più lunga di quella delle parole.

Una particolarità che si è evidenziata mentre si analizzava questo problema è legata al fatto che il l'algoritmo di Stanford eseguito in locale e sul server restituisce due risultati diversi. Originariamente l'algoritmo utilizzato era stato applicato mediante la connessione al loro server per velocizzare i processi, mentre in questa fase di controllo si è utilizzato l'algoritmo in locale perché si doveva provare il tagger su poche frasi e quindi la velocità non era una problema. Nonostante il modello utilizzato sia stato lo stesso in entrambi i casi, l'algoritmo in locale non spezzava alcuni di questi termini. Per effettuare un controllo rigoroso è stato quindi necessario utilizzare l'algoritmo con la connessione al server di Stanford. Questa particolarità dell'algoritmo non è riportata nella documentazione ufficiale in cui la connessione al server è considerata semplicemente una soluzione per velocizzare il processo di tagging [37]. A complicare ulteriormente la questione, si aggiunge un'ulteriore particolarità dell'algoritmo che per due parole applica comportamenti diversi a seconda del contesto. Le due parole sono *"dunno"* e *"gotta"* abbreviazione per *"going to"*. Talvolta a queste parole sono assegnati due tag distinti perché l'algoritmo applica la divisione mostrata in precedenza, ma in alcuni casi le parole sono tenute assieme e quindi si ottiene un unico tag come risultato. Non è stata compresa a pieno la logica

dietro a queste scelte, sicuramente se la parola è l'ultima della frase allora non viene spezzata, mentre se si trova in mezzo i comportamenti sono variabili. Per tutte le altre forme contratte se si trovano a fine della frase allora sono tenute assieme, altrimenti si ottengono due tag distinti.

4.2 Pattern

Identificati i problemi legati al tagger, il primo passaggio è stato quello della creazione di un dizionario con chiave il termine contratto e come valore i termini interpretati dall'algoritmo. Una funzione di controllo ha avuto il compito di ricostruire le frasi nello stesso modo in cui sono state interpretate dall'algoritmo di tagging. In aggiunta al dizionario sono state applicate delle eccezioni per gestire i casi speciali, come le parole candidate a fine frase in cui era necessario mantenere il valore della chiave del dizionario. Nella funzione di controllo sono stati immessi dei parametri opzionali per gestire quei pochi casi in cui la parola contratta era tenuta nella sua forma da parte del tagger; a seconda dei valori dei parametri la funzione di controllo avrebbe spezzato o meno il termine. Non conoscendo la logica dietro alla quale queste situazioni si verificavano, la funzione ha provato tutte le possibili combinazioni di valori fino a quando la lunghezza della lista delle parole coincidesse con quella delle tag.

Una volta sistemato il problema del tagging è stato possibile procedere con l'identificazione dei pattern di interesse. Secondo l'articolo di *Afzaal et al.* [16] gli utenti nelle loro recensioni non parlano esclusivamente di aspetti del film, ma è abbastanza comune trovare delle digressioni su altri argomenti. Queste porzioni della recensione non sono utili per identificare gli aspetti e sarebbe quindi ottimale tenere solamente le parti di reale interesse. Sempre secondo gli autori è possibile identificare gli elementi di interesse in base alla sequenza di tag delle parole.

4.2.1 Identificazione delle sequenze

Sempre secondo gli autori le sequenze di pattern rilevanti iniziano sempre con un nome e si concludono con un aggettivo o viceversa. Questa soluzione è molto sempli-

ficata però è necessario attuare questo passaggio per poi identificare quelle sequenze realmente di interesse. Per realizzare questa operazione si è creata una funzione che prende in input la lista delle recensioni e la corrispettiva lista di tag. I passaggi della funzione sono i seguenti:

- Sistemazione dei tag: si richiama la funzione della sezione precedente con il compito di far corrispondere la lunghezza delle parole con quella dei tag. Si provano tutte le combinazioni di valori dei parametri opzionali per massimizzare le probabilità di ottenere una corrispondenza.

- Sequenza di inizio: all'inizio di ogni recensione viene inizializzata una sequenza vuota che sarà aggiornata con i tag ritenuti idonei per formare la sequenza.

- Controllo dei tag: si scorrono i tag all'interno della lista, se la sequenza di inizio è vuota e si trova un nome o un aggettivo, quel tag viene designato come elemento di inizio della sequenza. Se la sequenza ha già un elemento di inizio si controlla se sia l'elemento conclusivo (un aggettivo se la sequenza inizia per nome o viceversa).

- Conclusione sequenza: se si trova il tag di termine sequenza essa viene salvata in una lista di output e si procede con il resto della frase. Conclusa la frase si ricomincia dal primo punto con una nuova sequenza.

L'algoritmo lavora sulle singole parole e non a livello di frase in quanto anche all'interno di una frase possono essere identificate più sequenze. Si prenda come esempio la frase *"The acting was good but it was a boring story."*, la necessità di analizzare le singole parole deriva dal fatto che all'interno di una singola frase possono coesistere più pattern. Nel caso riportato il primo pattern inizia con il nome *"acting"* e finisce con l'aggettivo *"good"* mentre il secondo inizia con l'aggettivo *"boring"* e termina con il nome *"story"*. Tutte le parole che non sono né un nome né un aggettivo sono aggiunte alla sequenza fino a quando non si arriva all'etichetta di stop.

Far corrispondere parole e tag era un obiettivo imprescindibile per questo passaggio perché in questo modo si identificano adeguatamente i pattern con le sequenze di parole. Gli autori del paper non specificavano le situazioni particolari nei casi in cui non fosse possibile identificare l'etichetta di stop. Se una sequenza inizia ma non è trovata l'etichetta corrispondente prima della fine della frase non verrà scartata ma

sarà salvata fino al punto finale della frase. Questa soluzione potrebbe introdurre del rumore con delle sequenze non del tutto pertinenti, ma una buona parte di esse sarà filtrata mediante il prossimo passaggio illustrato nella seguente sottosezione.

4.2.2 Identificazione dei pattern rilevanti

Come specificato dagli autori stessi del paper questo approccio è efficace ma restituisce un numero eccessivo di pattern diversi. Nel caso in questione, sia per le frasi con e senza spelling correction si sono ricavate oltre quattro milioni di sequenze rilevanti distinte. Per filtrare ulteriormente i risultati è necessario applicare un algoritmo di *GSP* (Generalized Sequential Pattern) o semplicemente scartare le sequenze meno frequenti. In *Tabella 4.1* sono riportati i numeri di pattern che si identificherebbero selezionando una percentuale di pattern più frequenti.

Pattern Frequenti	Tot. No Spelling	Tot. Spelling
0.001%	36.80%	36.74%
0.01%	50.09%	50.06%
0.1%	60.84%	60.81%
0.2%	63.67%	63.64%
0.5%	67.17%	67.14%
1%	69.66%	69.63%
2%	72.02%	71.99%
25%	81.46%	81.43%
50%	87.64%	87.62%

Tabella 4.1: Pattern più comuni in proporzione ai pattern totali

Come era lecito aspettarsi un numero ristretto di pattern costituisce la maggior parte dei pattern totali e la grande quantità di pattern identificati è da attribuirsi a dei pattern particolari che sono stati individuati pochissime volte all'interno dei commenti. Prendendo la prima metà dei pattern più comuni si ottiene l'87% dei

pattern totali sia per quelli con spelling correction che non. Le differenze sono molto più evidenti al calare della percentuale selezionata: ad esempio per il primo 1% si raggiunge quasi il 70%, la metà dei pattern totali può essere ricavata prendendo solamente lo 0.01%. Come specificato dagli autori del paper questo comportamento è comune in questo genere di analisi e solitamente si prende l'1% dei pattern più comuni. Dato che questo lavoro è incentrato su un numero molto più elevato di recensioni rispetto ad altri progetti, si è deciso di considerare lo 0.5% dei pattern più comuni raccogliendo circa i due terzi dei pattern totali.

La differenza fra pattern con e senza spelling correction è quasi nulla e aumenta con il calare dei pattern considerati. Come ultimo passaggio dai commenti si estraggono solamente quelli corrispondenti ai pattern più frequenti e si creano due nuovi oggetti: una lista di recensioni con solo i pattern di interesse e la relativa lista di tags, operazione realizzata sia per i commenti con e senza spelling correction. Queste due liste serviranno per il raggruppamento in diversi aspetti.

4.3 WordNet

A questo punto delle analisi, dopo aver standardizzato i commenti attraversoi il preprocessing, dopo aver individuato gli aspetti rilevanti da considerare, il passaggio successivo è quello di trovare in modo automatico degli aspetti da utilizzare come base per l'Aspect-Based Sentiment Analysis. Come riportato da diversi paper saranno i nomi che andranno a definire un aspetto piuttosto che un altro. A questo proposito è stata creata una funzione che prende in input le due liste ottenute al punto precedente, per ogni parola controlla se il tag di riferimento è quello di un nome, si effettua un controllo tramite una libreria specifica e se il controllo è superato il termine viene aggiunto ad un dizionario.

La libreria che effettua il controllo è WordNet [38], un database lessicale sviluppato dall'Università di Princeton [39] che permette di connettere fra loro parole in base al loro significato. Tra le tante funzioni, questa libreria permette di ottenere un sinonimo cognitivo di ogni parola, definito *synset*: se due parole hanno lo stesso sinonimo cognitivo allora rappresentano lo stesso elemento indipendentemente dal contesto. Questo synset sarà utilizzato più avanti per le tecniche di raggruppamen-

to, è quindi indispensabile che ogni termine abbia un sinonimo cognitivo, per questo motivo è stato utilizzato come funzione di controllo. Se un termine è stato identificato come un nome dal tagger, non è scontato che quella parola sia individuata anche all'interno di WordNet, se la parola non è presente è impossibile ricondursi al synset. Questi casi non sono molto rari in quanto il tagger, se non riconosce una parola, la classifica automaticamente come nome; WordNet permette quindi di ottenere un insieme di parole che sicuramente sono presenti in lingua inglese.

4.3.1 Similarità

Ricavato un insieme di nomi, è possibile calcolare la similarità tra due synset per determinare i termini simili fra di loro. La stessa libreria permette di scegliere fra diverse misure di similarità quella utilizzata in questo progetto sarà la Wu-Palmer [40]. La similarità fra due concetti *C1* e *C2* è calcolata come:

$$Sim_{wup}(C1, C2) = \frac{2N}{(N1 + N2)} \tag{4.1}$$

I synset all'interno di WordNet sono organizzati in una struttura ad albero che segue una determinata tassonomia. Questa struttura ad albero presenta nei nodi foglia i synset più specifici che spessso coincidono con la parola da cui sono stati ricavate e salendo di livello verso il nodo radice sono raccolti dei synset sempre più generali. Per esempio si considerino le parole *car* e *boat* i rispettivi percorsi da nodo foglia al nodo radice saranno: *car* $\rightarrow$ *automobile* $\rightarrow$ *vehicle* e *boat* $\rightarrow$ *vehicle*. Nell'equazione (4.1) i valori *N1* e *N2* sono la profondità dei percorsi prima di raggiungere un nodo comune rispettivamente per il primo e il secondo termine. Nel caso in esempio *car* ha un percorso più lungo per giungere a *vehicle*. Non è scontato che il nodo più profondo in comune sia un nodo radice da qui la presenza del termine N che rappresenta il livello di profondità del primo nodo in comune che è l'equivalente di $argmin(N1, N2)$.

L'insieme di valori che può assumere l'indice è $0 <= Sim_{wup} <= 1$ dove il valore minimo indica nessuna similarità, ovvero quando due synset non hanno nessuno nodo in comune ed appartengono a delle tassonomie diverse, mentre il valore massimo indica la presenza di sinonimi.

Per ogni coppia di nomi è stata calcolata la similarità e i risultati sono stati salvati in una matrice quadrata con oltre 20 mila righe. Per velocizzare il calcolo della similarità sono stati inseriti i valori solo nella matrice triangolare superiore e successivamente questi valori sono stati riportati, con le opportune inversioni, nella parte inferiore. Essendo una similarità l'ordine con cui i termini sono messi in input nella funzione non altera il risultato. In parallelo è stata costruita la lista nomi in modo tale da poter collegare i valori della matrice con i termini mediante un semplice indice.

4.4 Clustering

4.4.1 Clustering Gerarchico

Per il raggruppamento la tecnica utilizzata è stata quella del clustering, in particolare un clustering gerarchico in cui, come dice il nome stesso, si cerca di definire una gerarchia all'interno delle osservazioni. I cluster gerarchici seguono due possibili approcci:

- Agglomerativo: ogni osservazione parte come un cluster a sé e ad ogni step, secondo una funzione di link, i cluster sono raggruppati per formare un nuovo cluster. Il procedimento continua fino a quando non si arriva ad avere tutte le osservazioni in un unico cluster.
- Divisivo: si procede nella direzione opposta, partendo da un cluster unico con tutte le osservazioni, si individuano nelle diverse iterazioni cluster più piccoli fino ad arrivare a un singolo cluster per osservazione.

In questo progetto la tecnica utilizzata è stata quella agglomerativa, da qui in avanti tutti i riferimenti ai dettagli dell'algoritmo sono da intendersi per il metodo agglomerativo. Il link è una funzione, calcolata ad ogni step, per determinare quali siano i cluster candidati ad essere uniti. Dati due cluster r, s con le rispettive osservazioni all'interno x_r, x_s, le equazioni (4.2), (4.3) e (4.4) riportano le formule per calcolare tre diversi tipi di link: singolo, completo e medio:

$$L_{r,s} = \min(D(x_{ri}, x_{sj})) \tag{4.2}$$

$$L_{r,s} = \max(D(x_{ri}, x_{sj})) \tag{4.3}$$

$$L_{r,s} = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj}) \tag{4.4}$$

Nel caso di link singolo (4.2) i cluster candidati per essere uniti sono quelli definiti dalla distanza minore tra due osservazioni in due cluster diversi. Questo metodo funziona solamente con le osservazioni disposte in pattern particolari, come ad esempio un formato globulare, e generalmente non è indicato per la maggior parte degli scopi.

Il link completo (4.3) applica la stessa funzione ma nel senso opposto: la distanza fra due cluster è definita dalla distanza massima di due osservazioni al loro interno. Una volta calcolate tutte le possibili combinazioni di distanze fra i cluster, quelli con la distanza massima minore sono uniti. Questo approccio in letteratura ha delle performance migliori rispetto al link singolo, eccetto per dei pattern particolari nelle osservazioni.

Il link medio (4.4) calcola tutte le distanze possibili fra coppie di punti nei cluster e pondera il risultato per il numero di osservazioni totali. I cluster che hanno la distanza media minore saranno raggruppati. Questo approccio ha delle performance simili a quello completo, non esiste una regola ben precisa su quale dei due sia il migliore ma varia da lavoro a lavoro.

Come si evidenzia dalla *Figura 4.1* a seconda del link utilizzato la classificazione può variare anche in maniera sostanziale. Il link singolo funziona quando ci sono dei gruppi di osservazioni ben distinti fra di loro, una situazione che non si dovrebbe riscontrare nei dati utilizzati in quanto molti dei valori di similarità saranno lontani dai valori estremi. Gli altri due metodi sono più versatili, si adattano a diverse strutture di dati ma non è possibile dire a priori se uno dei due prevalga sull'altro.

Esiste un ulteriore tipo di link che è quello di Ward in cui i cluster più vicini sono definiti dalla minima somma delle differenze al quadrato fra tutte le coppie di punti. Questo metodo è applicabile solamente quando la matrice in input è una matrice di distanze euclidee, dunque non è applicabile su questo lavoro.

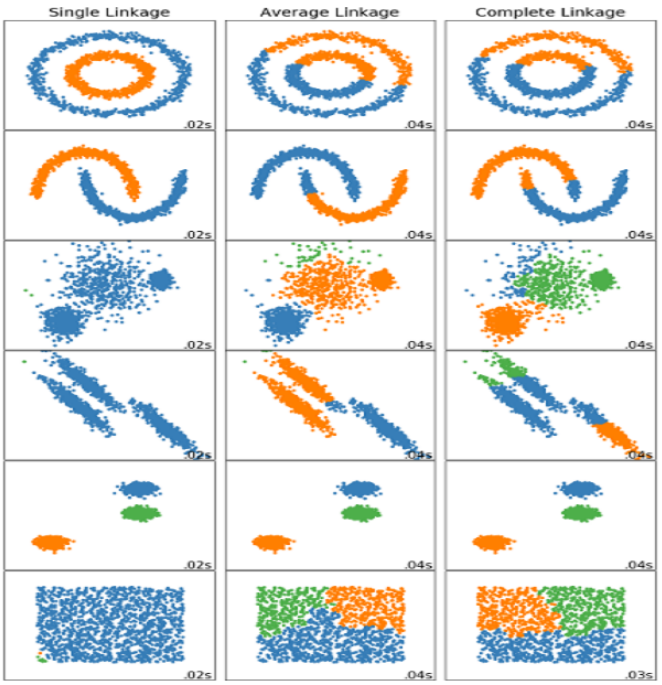

Figura 4.1: Comportamento del clustering al variare dei dati e del link su dataset di letteratura

Prima degli algoritmi di clustering sono necessarie delle modifiche alla matrice di similarità: i cluster gerarchici gestiscono tutte le matrici di distanze, anche quelle non euclidee, l'unico vincolo è che la metrica utilizzata sia una distanza. Nel caso in questione la matrice di similarità funziona nel modo esattamente opposto a quella delle distanze: termini simili hanno valori più alti, termini dissimili hanno valori bassi.

Le analisi sono state effettuate valutando un numero differente di cluster, provando tutte le tipologie di link disponibili e utilizzando la matrice delle distanze, ottenuta con il complementare ad uno della matrice delle similarità.

Per valutare la qualità del clustering è stato impiegato l'indice di Silhouette definito nell'equazione (4.5):

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}} \tag{4.5}$$

Le quantità *a(i), b(i)* sono calcolate come riportato nelle equazioni (4.6) e (4.7):

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C, i \neq j} d(i, j) \tag{4.6}$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \tag{4.7}$$

Il valore *a(i)* riferito all'osservazione i-esima è la distanza media di quella osservazione da tutte le altre all'interno del suo cluster C_i. Il valore *b(i)* è la distanza media minima della i-esima osservazione da tutte le osservazioni in un altro cluster. La prima quantità indica quanto una osservazione è coerente con il resto delle osservazioni nel suo cluster ed è ottimale che questo valore sia il più basso possibile, la seconda quantità indica quanto ben definita è una osservazione rispetto alle osservazioni del cluster più simile e dovrebbe essere il più grande possibile. I valori di *s(i)* si interpretano di conseguenza: più il valore è alto, più un'osservazione è simile a quelle del suo cluster e diversa dalle osservazioni del cluster più vicino, quindi il cluster riesce ad isolare bene un gruppo di osservazioni dal resto dei dati. Il termine al denominatore serve per prendere il valore massimo e fare in modo che l'indice sia compreso fra $-1 <= s(i) <= 1$ e facilitarne l'interpretazione. Questo valore di Silhouette è riferito ad una singola osservazione, il valore complessivo è dato dalla media dei singoli indice una volta terminato l'algoritmo di clustering.

4.4.2 HDBSCAN

Una seconda metodologia di clustering imlpementata è stata la versione gerarchica dell'algoritmo **DBSCAN** (Density-based spatial clustering of applications with noise), denominata **HDBSCAN**. L'algoritmo DBSCAN è stato proposto nel 1996 [41] e nel suo approccio identifica i cluster in base a delle zone di densità; l'algoritmo è uno dei più utilizzati e citati in letteratura. Selezionato un punto nello spazio, si definisce un parametro ϵ che identifica il raggio entro il quale si dovranno trovare altri punti simili. Se entro questo raggio si identifica un numero sufficiente di punti, allora le osservazioni formeranno un cluster. Si effettua la ricerca per tutti i punti non visitati e se alcuni punti non sono sufficientemente vicini ad altri o hanno un intorno di punti troppo poco numeroso, non sono classificati all'interno di nessun

cluster ma considerati come rumore. Il principale vantaggio di questo metodo è la possibilità di trovare cluster con geometrie diverse, non è richiesto il numero di cluster a priori ed esiste il concetto di punto rumore. Gli svantaggi sono la necessità di specificare accuratamente il parametro ϵ e i problemi nella gestione di aree con densità molto diversa.

HDBSCAN è la trasposizione gerarchica di questo metodo che si basa sul concetto di stabilità del cluster. La variazione più importante rispetto al metodo originario è l'assenza del parametro ϵ e l'aggiunta di un numero minimo di osservazioni per poter considerare un gruppo di dati di un cluster. L'algoritmo ha lo stesso scopo finale, quello di trovare delle zone ad alta densità ed eventualmente delle zone con rumore. Gli step dell'algoritmo sono:

1. Per ogni osservazione si calcola la *core distance* la distanza di quel punto dal k-esimo vicino più vicino.

2. Si calcola la *mutual reachability distance* per ogni osservazione come riportato nell'equazione (4.8).

3. Le osservazioni sono considerate come un grafo pesato in cui i punti sono i vertici e la distanza calcolata al punto precedente è il peso assegnato ai nodi. Il grafo viene spezzato progressivamente in base ad un valore soglia molto basso che progressivamente aumenta. Si parte da uno step iniziale con un grafo connesso a delle osservazioni singole.

4. Questo processo può essere rappresentato come un dendrogramma in cui si può visualizzare a quale distanza ciascuna osservazione è stata unita ed è possibile calcolare il valore $\lambda = \frac{1}{distance}$ che sarà la base del criterio con cui si sceglierà il numero di cluster in output.

Per quantificare la distanza fra le osservazioni è possibile prendere il valore della matrice delle distanze oppure tramite la *mutual reachability distance* si può creare una regola tramite la quale si pondera la distanza di due osservazioni e il numero di k vicini che possiedono:

$$d_{mr-k}(a,b) = \max\{core_k(a), core_k(b), d(a,b)\} \tag{4.8}$$

Dove $core_k(a)$ è la *core distance* di a rispetto al k-esimo vicino più vicino. In

questo modo se due osservazioni sono molto vicine ma hanno i vicini molto lontani, viene considerato questo valore come riferimento. Con questa strategia si cerca di enfatizzare l'importanza delle osservazioni con vicini molto stretti per creare delle piccole zone ad alta densità.

Per ogni cluster è possibile definire un valore λ_{birth} e λ_{death} in base a quando viene creato e diviso, inoltre è possibile ricavare il valore λ_p per ogni osservazione. Per ogni cluster è calcolato il valore di stabilità come riportato nell'equazione (4.9):

$$Stability_c = \sum_{p \in c} (\lambda_p - \lambda_{birth}) \tag{4.9}$$

Partendo dalla base, dalle singole osservazioni, si calcola la stabilità per ogni cluster temporaneo che viene a crearsi: nel caso in cui i cluster figli abbiano una somma di stabilità più alta del cluster in cui sono raggruppati, questa somma viene presa come riferimento per i futuri raggruppamenti, al contrario se il nuovo cluster ha una stabilità maggiore è considerato un cluster migliore e la sua stabilità è la nuova soglia di riferimento. Una volta arrivati alla radice si individuano i valori identificati durante il percorso e in questo modo si ricaverà il migliore insieme di cluster. Con questo sistema è possibile identificare delle osservazioni che non saranno mai raggruppate e corrisponderanno al rumore.

4.4.3 Word2vec

I modelli Word2vec, pubblicati per la prima volta da un gruppo di ricerca di Google nel 2013, sono una classe di algoritmi che mappano le parole in uno spazio multidimensionale a valori numerici. Con questi algoritmi è possibile offrire un nuovo approccio per l'identificazione degli aspetti con particolare attenzione agli aspetti latenti. Esistono due tipologie di modelli Word2vec:

- Modelli da addestrare: un corpus di documenti viene dato in input al modello. Il modello è una rete neurale a due strati, già allenati per ricostruire il contesto linguistico delle parole, che, dopo diverse iterazioni, restituirà come risultato un vettore numerico di lunghezza predefinita per ciascuna parola.

- Modelli già addestrati: sono modelli già pronti per l'utilizzo e addestrati su un corpus esterno di dati.

A seconda del corpus di documenti e di parametri del modello la rappresentazione di una parola può variare da modello a modello. L'idea comune è però sempre quella di fare in modo che in un'ipotetica rappresentazione di questo spazio a tantissime dimensioni le parole simili fra di loro si trovino vicine e quelle diverse agli antipodi. Il vantaggio di utilizzare un modello già addestrato è quello di risparmiare tempo per la creazione del modello, avere una rappresentazione precisa anche di termini poco frequenti, i corpus disponibili in letteratura contengono milioni di frasi, però può esserci lo svantaggio di avere uno spazio che non rappresenti bene l'ambito in cui verrà applicato il modello. Per avere un'idea più precisa delle potenzialità del modello, in questo progetto sono stati implementati entrambi gli approcci.

Per l'addestramento del modello possono esserci due strategie:

- CBOW: Continuous Bag of Words, si prova a predire la parola dato il contesto intorno ad essa.

- Skip-gram: partendo da una parola si cerca di identificare il contesto in cui è inserita.

Il primo metodo è più veloce ed ha una buona accuratezza per i termini frequenti, il secondo è più lento, funziona bene anche con pochi dati e con termini rari.

Per quanto riguarda il modello già addestrato è stato preso come riferimento uno dei primi modelli proposti da Google, addestrato su oltre cento miliardi di parole e contiene tre milioni di vettori per parole e frasi. I corpus di riferimento erano delle notizie tratte da Google News, l'ambito in cui è stato addestrato questo modello non è lo stesso del dataset ma può essere coerente in quanto le notizie erano generiche e ricoprivano diversi ambiti tra cui anche il cinema e lo spettacolo.

Il modello addestrato sui dati ha utilizzato i seguenti parametri:

- Strategia: si è preferito la tecnica CBOW in quanto più veloce e accurata per i termini più comuni che sono stati considerati come la parte più importante per il raggruppamento rispetto al riconoscimento di termini più rari.

- Finestra: questo parametro indica quante sono le parole da considerare per le co-occorrenze in una finestra utilizzata dal modello durante il processo di

apprendimento. Una finestra più ampia permette di cogliere aspetti più diversificati ma è più dispersivo. Si è tenuto il parametro di default, cinque, utilizzato anche nel modello già addestrato.

- Frequenza: la frequenza minima di una parola per poter avere una sua rappresentazione vettoriale. Non esiste una regola precisa su quale sia un valore corretto per questo parametro ma è fortemente dipendente dal contesto, in questo progetto si è selezionata una soglia minima di cento occorrenze, un numero abbastanza elevato che è stato suggerito dai risultati ottenuti con le tecniche precedenti.

- Dimensione del vettore: è stata fissata a 300 come per il modello già addestrato. Un vettore più grande permette di diversificare l'informazione fra i termini ma si rischia di forzare l'algoritmo ad assegnare valori nel vettore se la dimensionalità è eccessiva. Da studi di letteratura è risultato che le performance migliorano fino al raggiungimento di questa dimensione e non migliorano, in alcuni casi peggiorano, con l'aumentare.

- Iterazioni: il numero di iterazioni che deve compiere la rete neurale per creare il modello. Il parametro è stato fissato a 30 iterazioni, anche questo un numero consigliato in letteratura. La maggior parte del guadagno lo si ottiene nelle prime iterazioni con un significativo miglioramento del modello; superata una certa soglia di iterazioni, solitamente il modello inizia a peggiorare in quanto si raggiunge il limite della tecnica e l'algoritmo procede con un addestramento eccessivo che non porta benefici.

Dato che la rappresentazione delle parole è differente rispetto agli altri metodi, è necessario definire una nuova metrica di similarità per costruire la matrice. La metrica già presente nel pacchetto di Word2vec è la *similarità coseno*. Dati due vettori $\mathbf{A}$ e $\mathbf{B}$ la similarità coseno è definita in formula (4.10) come il coseno dell'angolo che si forma fra gli elementi che definiscono i punti nello spazio multidimensionale:

$$sim = \cos(\theta) = \frac{\mathbf{A} \times \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{4.10}$$

Dove A_i e B_i sono gli i-esimi elementi dei vettori. Dal valore dell'orientamento dei due vettori si calcola il coseno per renderlo una similarità. Il valore della similarità coseno varia da un massimo di 1 che si ottiene quando l'angolo fra i due vettori è nullo, dunque i vettori sono identici e orientati nella stessa maniera, fino a un minimo di -1 che è il caso in cui i vettori hanno orientamento opposto e si genera un angolo piatto. In mezzo ci sono tutte le sfumature che definiscono la similarità fra i due vettori, il cui valore cala al crescere dell'angolo.

4.5 Risultati della Cluster Analysis

Per le tecniche dove la matrice di similarità è stata costruita mediante WordNet, nel 66% dei pattern ritenuti rilevanti sono stati ricavati 19701 nomi distinti, ognuno dei quali rappresentava una riga e una colonna della matrice di similarità. I risultati per il clustering agglomerativo sono rappresentati in *Figura 4.2*.

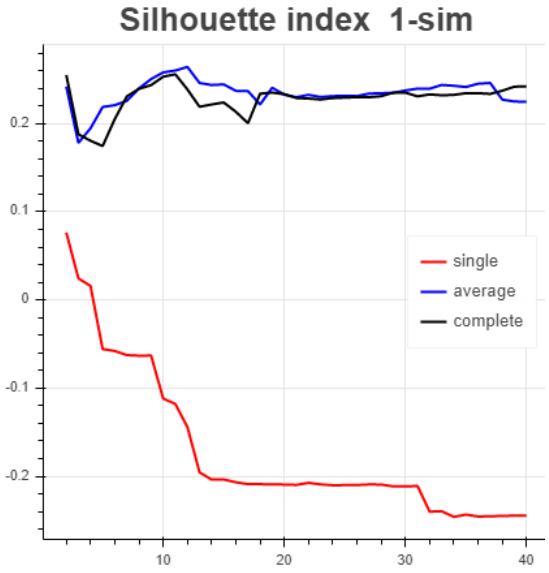

Figura 4.2: Indice di Silhouette nel clustering agglomerativo

Cluster Id	#Osservazioni
1	6734
2	2316
3	2042
4	1047
5	1012
6	787
7	510
8	492
9	433
10	379
11	274
12	2

Tabella 4.2: Cluster e numerosità

Sull'asse delle ascisse sono riportati i numeri di cluster creati e sulle ordinate il relativo indice di Silhouette, dove un valore maggiore indica migliori performance. Come era prevedibile il link singolo non ottiene delle buone prestazioni e si attesta sempre sotto gli altri due metodi con una differenza che aumenta all'aumentare del numero di cluster. Gli altri due metodi di link forniscono dei risultati pressoché identici, con l'andamento delle due spezzate che in alcuni punti si sovrappone. Sono stati provati gli algoritmi da un minimo di due cluster, anche se poco informativi, fino ad un massimo di quaranta. Con solamente due cluster l'algoritmo restituisce valore soprendentemente elevati, l'indice cala fino a cinque cluster per poi risalire e raggiungere il massimo valore intorno a 0.26 in prossimità di dodici cluster con il link medio. Il link medio raggiunge il massimo con undici cluster con un valore di 0.25. I valori dell'indice per questi due tipi di link si stabilizzano intorno a 0.23 a partire dai venti cluster e proseguono sostanzialmente inalterati fino al numero massimo considerato.

Uno dei problemi di questa classificazione è la numerosità dei cluster ottenuti come riportato nella *Tabella 4.2*: il primo cluster per dimensioni ha quasi il triplo delle parole del secondo più numeroso. Questo notevole squilibrio nelle dimensioni non è necessariamente legato a una classificazione scadente, ma osservando le etichette assegnate alle parole si è notato che molti dei termini che avrebbero dovuto indicare un aspetto sono stati in realtà inseriti in un unico cluster. Anche analizzando gli output di altri algoritmi vicini alle performance del migliore, si è riscontrato questo problema.

Una prima semplificazione dell'algoritmo è stata con la rimozione dei sinonimi all'interno della matrice. Le coppie di parole con una similarità Wu-Palmer pari all'unità sono state considerate sinonimi e di queste coppie si è tenuto solo il termine più frequente togliendo le altre considerabili come informazioni superflue e già contenute nel termine più comune. Con questo approccio sono stati rimossi circa 4000 termini ma le performance del clustering agglomerativo non sono variate con i risultati che rispecchiavano quello già ottenuto in precedenza.

Nell'effettuare la clustering analysis fino a questo punto si sono considerati tutti i nomi nei pattern rilevanti, anche quelli che compaiono con una bassissima frequenza. Un ulteriore step per semplificare l'algoritmo è stato quello di considerare i mille termini più comuni all'interno di tutti i pattern rilevanti e di considerare una matrice ridotta relativa ai nomi più frequenti. Anche in questo caso sono stati

rimossi i sinonimi per evitare ridondanze, dunque la matrice quadrata non conteneva esattamente mille termini ma 849. I valori si Silhouette relativi al clustering agglomerativo con questa matrice sono presentati in *Figura 4.3*:

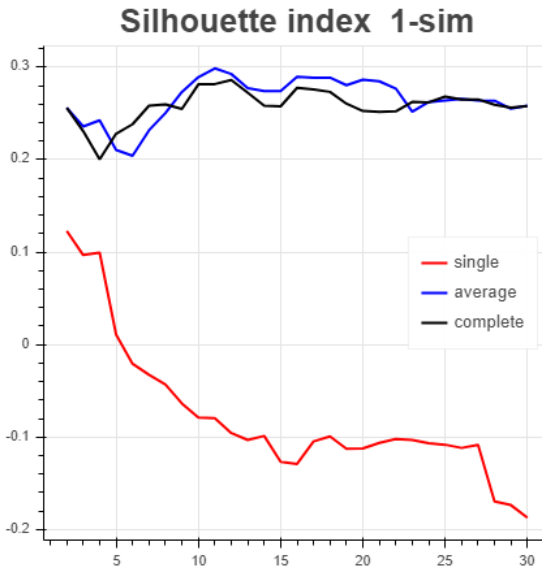

Figura 4.3: Indice di Silhouette con i nomi più comuni

Cluster Id	#Osservazioni
1	159
2	152
3	145
4	104
5	99
6	52
7	42
8	35
9	32
10	12
11	10
12	7

Tabella 4.3: Cluster e numerosità

Il grafico presenta sempre l'indice di Silhouette sulle ordinate e il numero di cluster considerato nell'algoritmo sulle ascisse, in questo caso il valore massimo è stato limitato ad una trentina di cluster, considerando anche la dimensione ridotta della matrice. L'andamento dell'indice è pressoché identico a quello ottenuto con la matrice intera senza rimozioni: il link singolo è il peggiore, mentre i link medio e completo hanno le stesse performance. Anche in questo caso l'indice ha un alto valore per soli due cluster, scende rapidamente e a partire dall'algoritmo con sei cluster inizia a salire fino a raggiungere i valori massimi ancora una volta nell'algoritmo con link medio e 12 cluster.

Come riportato nella *Tabella 4.3* con questo algoritmo non solo si raggiungono delle performance leggermente migliori, indice di Silhoutte pari a 0.3, ma anche la distribuzione dei termini nei cluster è molto più bilanciata rispetto al primo modello.

Per cercare di migliorare ulteriormente le performance della cluster analysis, si è provato a creare una variante della matrice di similarità. La similarità di Wu-

Palmer utilizzata in precedenza calcolava il valore di similarità per qualsiasi coppia di parole, anche nei casi in cui i nomi non avevano un albero delle gerarchie in comune. L'algoritmo forzava la creazione di un albero fittizio per poter connettere i due termini e stimare una loro similarità. Il motivo per cui è stato applicato questo approccio era legato alla possibilità di ottenere una matrice sparsa, principalmente composta da zeri, e quindi di non avere sufficienti informazioni da dare all'algoritmo di clustering per poter differenziare i termini.

L'applicazione della nuova matrice ha comportato un miglioramento non significativo per quanto riguarda la matrice completa e con i nomi più comuni il risultato era identico.

Per il metodo HDBSCAN le analisi hanno seguito lo stesso percorso partendo con una matrice completa per poi passare alla matrice dei termini più comuni. Le principali difficoltà riscontrate con questo metodo sono state relative all'impossibilità di ottenere dei cluster ben bilanciati in termini di dimensioni. Il problema, già presente nel clustering agglomerativo, è stato molto più marcato con questa tecnica dove in alcuni casi fino al 80% delle osservazioni ricadevano nel cluster principale, come riportato nella *Tabella 4.4*.

L'algoritmo prende in input un parametro relativo al numero minimo di osservazioni necessarie per creare un cluster e il numero di cluster che si ottiene non è definibile in input. Come si nota dai valori della terza colonna, il problema della dimensione è sempre presente indipendentemente da quante osservazioni sono richieste per la creazione di un cluster e dal numero di cluster identificati. Come conseguenza di questo comportamento anche l'indice di Silhouette ha valori molto più bassi che si aggirano intorno allo zero.

Anche considerando i termini più comuni l'algoritmo produce gli stessi output senza ovviamente migliorare i valori di Silhouette. Con meno osservazioni è stato possibile visualizzare una rappresentazione della logica che l'algoritmo ha seguito per la creazione dei cluster, come rappresentato nella *Figura 4.4*.

L'immagine fa riferimento al modello migliore trovato con la matrice ridotta, il parametro in ingresso erano almeno cinque osservazioni per creare un cluster e il numero di cluster trovati è stato tre. Si può notare come il grafico si sviluppi tutto da una singola parte e molte delle osservazioni sono staccate dal resto del grafico. I cerchi rappresentano i cluster identificati, in questo caso se ne notano solo due, uno blu ed uno arancione che è talmente piccolo da essere rappresentato semplicemente solo come una linea. Il terzo cluster, quello mancante, è delle osservazioni considerate

Minimo osservazioni	#Cluster	Dimensione
10	15	15187
20	10	15206
30	8	15206
40	7	15206
50	6	15275
60	6	15275
70	4	15412
80	4	15412
90	4	15412
100	3	15535
110	3	14888
120	3	14888
130	3	14888
140	3	14888
150	3	14888

Tabella 4.4: HDBSCAN: numero di cluster, dimensione al variare del numero minimo di osservazioni

come rumore, che sono troppo diverse dal resto dei dati per poter essere incluse in un cluster. Anche modificando i parametri del modello non si è riuscito ad ottenere un risultato migliore di quello presentato e di conseguenza questo algoritmo non sarà considerato nelle prossime analisi.

Una delle possibili spiegazioni è che nella matrice di similarità non si creano molte zone di densità e quindi il punto principale di questo algoritmo viene a mancare [42]. La creazione di un unico insieme molto compatto di osservazioni può anche spiegare il perché nel clustering agglomerativo il link singolo non si discostava più di tanto dagli indici di Silhouette degli altri due link per quanto riguarda la creazione di due cluster per poi calare drasticamente all'aumentare dei cluster.

Anche i cluster ricavati dai modelli word2vec non hanno ottenuto dei buoni risultati per gli indici di Silhouette. Con questa tecnica la matrice di similarità ha assunto delle dimensioni differenti in quanto molti più termini sono stati riconosciuti all'interno del modello. Con il modello di Google, la matrice era quasi un terzo più grande e conteneva quasi 30 mila termini. Questa grande quantità di nomi non ha

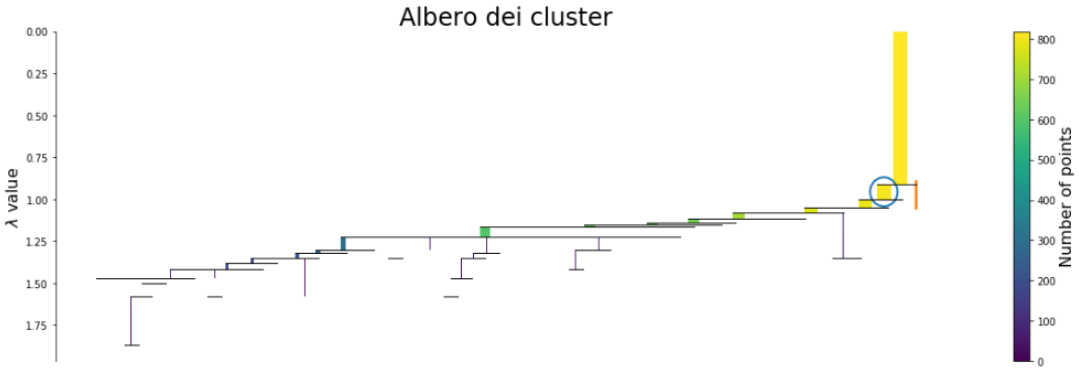

Figura 4.4: Generazione dei cluster con l'algoritmo HDBSCAN

però aiutato la clusterizzazione che è risultata, in termini di indice di Silhouette, al pari del metodo HDBSCAN. Anche tenendo solamente le parole più frequenti non si è notato un miglioramento sostanziale. Mentre l'andamento dell'indice al variare del numero di cluster aveva un andamento simile a quelli visti in precedenza, con i valori migliori per un numero ristretto di cluster, considerando solamente i termini più comuni i modelli migliori risultavano quelli con un numero di cluster elevato. La stessa identica situazione si è presentata sul modello creato basandosi sul dataset delle recensioni, con performance leggermente migliori di HDBSCAN ma molto lontane da quelle del clustering agglomerativo. Una possibile spiegazione di questi risultati può essere relativa alla costruzione del modello, alla scelta dei parametri anche se sono state utilizzate delle combinazioni di valori considerate buone dalla letteratura. Le similarità fra i due modelli escludono anche una dipendenza del contesto, il primo modello è stato creato facendo riferimento non solo ai film, potrebbe quindi trattarsi semplicemente di un caso in cui una tecnica, il clustering agglomerativo, semplicemente funziona meglio di altre. Un ultimo tentativo per migliorare questi modelli è stata l'applicazione di una fase di lemmatizzazione, ma non è stata poi realizzata in quanto dalla dimensione della matrice di similarità si è notato che la dimensione variava nell'ordine di poche decine di termini e dunque non si è ritenuto necessario creare altri modelli che verosimilmente avrebbero ottenuto gli stessi identici risultati.

Le stesse analisi sono state replicate sul dataset in cui le recensioni sono state soggette alla spelling correction. I risultati sono in linea con quelli ottenuti sull'altro dataset con il clustering agglomerativo che si conferma il metodo migliore seguito dal modello word2vec ed infinde HDBSCAN. A livello di valori dell'indice di Silhouette

non si è riscontrata una differenza rilevante: per il clustering agglomerativo nella matrice con tutti i termini i risultati erano leggermente migliori ma quasi uguali per quella con le parole più comuni. Quest'ultimo è risultato ancora una volta il migliore di tutti in termini di indice di Silhouette. Miglioramenti più evidenti si sono registrati per il modello word2vec, ma il divario con il clustering agglomerativo rimaneva ampio.

Capitolo 5

Modelli di classificazione

Prima di addestrare i modelli di classificazione è stato necessario individuare una tecnica che permettesse di connettere i cluster e le parole ottenute nella fase precedente con i pattern rilevanti e conseguentemente l'assegnazione di una classe. Dai 12 cluster identificati nel modello migliore, si è provato a raggrupparne alcuni in quanto non erano abbastanza numerosi o non era possibile identificare un aspetto concreto. Dalle successive analisi è però risultato più conveniente a livello di prestazioni mantenere la numerosità suggerita dall'algoritmo senza effettuare delle ulteriori modifiche.

Il dataset con i termini più comuni è stato arricchito con una nuova colonna contenente i sinonimi individuati dal modello di WordNet; questi termini saranno considerati al pari della parola di riferimento in termini di classe. Dal dataset delle informazioni si è ricavata la lista di attori e personalità legate ad uno specifico film. Il dataset originale conteneva per ogni riga un singolo film e una singola lista di personalità; per semplificare il processo di assegnazione delle classi si è realizzato un match fra il dataset delle recensioni e quello delle informazioni. Il risultato è un nuovo dataset in cui ad ogni recensione è affiancata la lista delle personalità.

Dai risultati della clusterizzazione, dal dataset dei pattern e questo delle informazioni è stata creata una funzione di assegnazione delle classi. Il funzionamento si può riassumere nei seguenti punti:

- Si è iterato all'interno di ogni recensione, all'interno di ogni frase e all'interno di ogni pattern rilevante.
- Si è controllato se esistesse effettivamente un pattern per quella frase, in caso affermativo si è controllato se fosse presente il nome di una personalità al

suo interno. Nel caso fosse stato identificato almeno un nome, il pattern era classificato automaticamente con una nuova classe, riservata esclusivamente ai nomi delle personalità.

- Nel caso non fosse presente un nome di un attore si itera per ogni termine all'interno del pattern. Nel caso in cui una di queste parole fosse presente all'interno del dataset della clusterizzazione, si aggiungeva ad una lista l'etichetta del cluster corrispondente. Questa operazione è stata realizzata anche sulle parole classificate come sinonimi, prendendo come riferimento l'etichetta della parola a cui erano collegate. Nel caso in cui la parola non fosse presente né nei termini utilizzati per la cluster analysis né all'interno dei sinonimi, alla lista era aggiunto un valore fisso di base per indicare l'assenza di quel termine nel dataset.

- Identificato un termine si ricavava anche la sua frequenza, salvato in una variabile a parte. Questa informazione sarà utile successivamente.

- Analizzate tutte le parole all'interno di un pattern le etichette delle parole non utilizzate per la cluster analysis sono state rimosse e si è effettuato un conteggio delle rimanenti.

- Una lista vuota significa che tutti i termini del pattern non sono stati utilizzati per la cluster analysis e quindi non è possibile realizzare un'assegnazione. Questi pattern rappresenteranno una nuova classe, quella dei pattern generali.

- Nel caso in cui la lunghezza della lista non sia nulla, il pattern viene classificato in base all'etichetta più frequente al suo interno.

- Nel caso in cui ci siano più etichette con massima frequenza il pattern viene assegnato alla classe corrispondente al termine con la frequenza minore. In questa maniera si cerca di valorizzare la presenza di termini poco frequenti all'interno dei pattern.

L'output di questa funzione è un oggetto stratificato con diversi livelli di liste che rispecchia la struttura del dataset delle recensioni. Le liste sono state condensate fra di loro per creare un'unica lista escludendo tutti quei pattern che non presentavano nessuna parola. Da questa lista si è creato un nuovo dataset in cui ogni riga rappresenta un singolo pattern e la classe di appartenenza.

Questo dataset è stato diviso in train e test set come avviene comunemente applicando una divisione tramite una stratificazione delle classi, in modo da mantenere le proporzioni delle classi sia all'interno del train set che nel test set, e utilizzando un seme fisso per riprodurre la stessa divisione per tutti i modelli. Sono stati utilizzati valori classici di divisione con il train set contenente il 70% delle osservazioni e il test set il restante 30%.

Dovendo lavorare esclusivamente su dei testi, non si hanno a disposizione delle variabili numeriche per addestrare i modelli ed è quindi necessario trovare un metodo per trasformare i testi in un oggetto analizzabile dai modelli. La tecnica scelta è quella del **TF-IDF** (*Term frequency - Inverse Document Frequency*. Questa variabile ha il compito di esprimere l'importanza di una parola relativamente ad un corpus di documenti. Il valore aumenta più una parola compare all'interno di un documento ma diminuisce con il numero di documenti in cui essa è presente. Questo valore tende a premiare quei termini che hanno un elevata frequenza ma il loro utilizzo è limitato ad un numero ristretto di documenti, in questo caso pattern. La formula per calcolare il TF-IDF è presentata nell'equazione (5.1):

$$TFIDF_{t,d} = tf(t,d) \times idf(t) \tag{5.1}$$

Dove $tf(t,d)$ rappresenta la frequenza del termine t all'interno del documento d e $idf(t)$ è la frequenza inversa del termine all'interno dei documenti. Mentre la frequenza di un termine all'interno di un documento è semplicemente un conteggio di quante volte il termine compare all'interno di un dato documento, la seconda quantità non ha una formula fissa ma esistono diverse alternative per il calcolo. La libreria utilizzata applicava la formula: $idf(t) = log(\frac{n}{df(t)})$, dove n rappresenta il numero totale di documenti. Il risultato è un vettore per ogni parola con un valore numerico che rappresenta l'importanza della parola stessa. Questo valore deve essere centrato, tramite una semplice operazione di standardizzazione, sottraendo un valore comune μ e dividendo per una varianza comune σ. Questi valori sono ricavati partendo dai valori di TF-IDF sul train set e sono applicati anche sul test set in modo da mantenere le proporzioni e applicare correttamente i classificatori.

Nelle seguenti sezioni saranno presentati i modelli di classificazione utilizzati.

5.1 Multinomial Naive Bayes

Il **Multinomial Naive Bayes** è una variante del classificatore Naive Bayes applicato su più di due categorie. È uno dei modelli più semplici e veloci a livello computazionale, ha diversi problemi strutturali di impostazione come sottolineato da *Rennie et al.* (2003) [43] ed è solitamente considerato il classificatore baseline da prendere come base per l'interpretazione della qualità dei risultati. Come suggerisce il nome questo modello sfrutta il teorema di Bayes per calcolare le probabilità di appartenenza alle classi e di conseguenza nelle analisi dei testi con questa tecnica si suppone l'indipendenza di ogni parola rispetto a tutte le altre all'interno di un documento, fatto che nella realtà, non si verifica mai. Alcuni studi hanno cercato di spiegare il perché questa tecnica ottiene comunque delle performance discrete nonostante la condizione di indipendenza non sia minimamente rispettata. Nel paper di *Zhang* (2004) [44] è riportato che una possibile spiegazione deriva dal fatto che le dipendenze esistono all'interno di ogni classe e per questo motivo si bilanciano fra di esse, inoltre con determinate combinazioni di dipendenze, l'effetto di una potrebbe bilanciare la dipendenza presente in un'altra classe.

Il modello, dato un insieme di classi $c \in 1, 2, \ldots, m$, stima un vettore $\theta_c = (\theta_{c1}, \theta_{c2}, \ldots, \theta_{cn})$ per ogni classe c in cui n è la dimensione del vocabolario, θ_{ci} la probabilità che l'i-esimo termine appaia nella classe c con la somma del vettore delle probabilità pari ad uno per ogni classe, $\sum_i \theta_{ci} = 1$. Da questi valori si può stimare la probabilità di un documento che è il prodotto dei parametri delle parole al suo interno come mostrato nell'equazione (5.1):

$$P(d|\theta_c) = \frac{(\sum_i f_i)!}{\prod_i f_i!} \prod_i (\theta_{ci})^{f_i} \tag{5.2}$$

dove f_i è la frequenza dell'i-esimo termine del documento. Assegnando delle probabilità a priori, è possibile dimostrare [45], che il classificatore con l'errore minimo è rappresentato dalla formula nell'equazione (5.2):

$$l(d) = \arg\max_c \left[\log p(\theta_c) + \sum_i f_i \theta_{ci} \right] \tag{5.3}$$

Mentre il numero di classi e la numerosità degli elementi al loro interno è definita, i parametri devono essere stimati dai dati di training. Questi parametri sono stimati

considerando il numero di volte che l'i-esimo termine compare nella classe c, N_{ci}, e il numero totale di termini nella classe c, N_c. Nella formula sono aggiunti dei valori α_i di smoothing per aumentare la numerosità dei termini, solitamente posto pari a uno, per gestire dal punto di vista matematico le operazioni per quei termini che compaiono solo nel test set e non nel train. L'equazione (5.3) riporta la stima di un singolo parametro:

$$\hat{\theta}_{ci} = \frac{N_{ci} + \alpha_i}{N_c + \alpha} \tag{5.4}$$

dove α è la somma di tutti gli α_i. Sostituendo questa quantità nell'equazione (5.2) si ottiene il metodo in cui il Multinomial Naive Bayes effettua la classificazione.

5.2 Complement Multinomial Naive Bayes

Il **CMNB** (*Complement Multinomial Naive Bayes*) è un miglioramento del modello Multinomial Naive Bayes proposto da *Rennie et al.* (2003) [43] per superare i problemi classici del classificatore nella sua forma base. Per stimare la probabilità di appartenenza di una parola ad una classe, a differenza del metodo di base, la verosimiglianza è stimata prendendo in considerazione il numero di volte che la parola compare in tutte le altre classi. In questo modo si hanno più dati per stimare il vettore di parametri e si dovrebbe raggiungere un livello di bias più basso. L'equazione (5.4) rimane la stessa ma al posto di N_{ci} e N_c saranno utilizzati rispettivamente $N_{\bar{c}i}$ e $N_{\bar{c}}$ per indicare il complementare alla classe c. L'equazione (5.5) mostra come si modifica il calcolo dell'errore minimo dell'equazione (5.3):

$$l(d) = \arg\max_c \left[\log p(\theta_c) - \sum_i f_i \theta_{ci} \right] \tag{5.5}$$

L'unica variazione è il segno meno al posto del più poiché l'assegnazione deve essere fatta alla classe che peggio corrisponde al complementare delle altre classi. Se prima un valore alto del secondo addendo indicava un'alta verosimiglianza di appartenenza

alla classe, adesso con il complementare un valore basso indica che una specifica parola è poco presente in tutte le altre classi ed è quindi l'oggetto da ricercare.

Secondo gli autori, questa modifica permette di mantenere i vantaggi del Naive Bayes, prima di tutto la velocità computazionale, migliorando significativamente le performance.

5.3 Support Vector Machine

La **SVM** (*Support Vector Machine*) è un algoritmo supervisionato di machine learning considerato uno dei migliori metodi per i task di classificazione dei testi. L'idea di base è quella di mappare i punti in uno spazio multidimensionale e trovare degli iperpiani che dividono questo spazio per separare gli elementi appartenenti a classi diverse [46]. In uno spazio multidimensionale è possibile creare infiniti iperpiani, l'algoritmo li seleziona in modo tale che la distanza dei punti, definita margine, dell'iperpiano che identifica due sottopiani di classi, sia la più grande.

Per quanto riguarda la classificazione binaria, il problema si riduce all'ottimizzazione della funzione riportata nell'equazione (5.6):

$$\min_{\mathbf{w}\in H, b\in\mathbf{R}\xi_i\in\mathbf{R}} \left(\frac{1}{2}\mathbf{w^T w} + C\sum_{i=1}^{l}\xi_i \right) \tag{5.6}$$

Dove $\mathbf{w}$ è un vettore di pesi, C è un parametro di penalizzazione che serve per premiare i modelli più semplici evitando di incorrere in problemi di overfitting e $\xi_i = \max(0, 1 - y_i(wx_i - b))$, soggetta a due vincoli: il primo di non negatività e il secondo $y_i(\mathbf{w^T}\varphi(\mathbf{x_i}) + b) \geq 1 - \xi_i$, con φ funzione che sarà trattata più avanti.

Nel caso di più di due classi [47] l'equazione (5.6) è riscrivibile come:

$$\min_{\mathbf{w_m}\in H, \mathbf{b}\in\mathbf{R}^k\xi_i\in\mathbf{R}^{l\times k}} \left(\frac{1}{2}\sum_{m=1}^{k}\mathbf{w_m^T w_m} + C\sum_{i=1}^{l}\sum_{t\neq y_i}\xi_{i,t} \right) \tag{5.7}$$

Con la conseguente funzione decisionale [48] rappresentata da:

$$\arg\max_m f_m(\mathbf{x}) = \arg\max_m(\mathbf{w_m^T}\varphi(\mathbf{x}) + b_m) \tag{5.8}$$

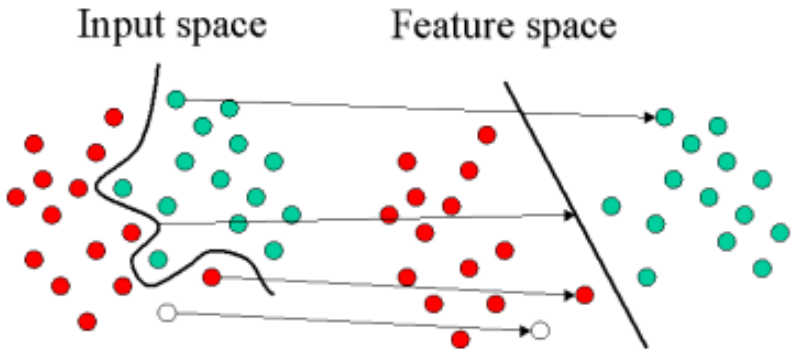

Figura 5.1: Trasformazioni delle funzioni kernel

Nella maggior parte dei casi non è possibile separare tutti gli oggetti di classi differenti utilizzando semplicemente un iperpiano lineare, in quanto la disposizione dei dati nell'iperspazio è solitamente molto complessa. Per risolvere questa limitazione è possibile creare una rappresentazione diversa dei dati in input mappandoli in un altro spazio in cui poi è possibile trovare un iperpiano che li divida perfettamente come riportato nella *Figura 5.1*.

L'elemento $\varphi(\mathbf{x})$, citato in precedenza, è una funzione matematica, definita funzione kernel in questo contesto, con la quale si effettua una trasformazione dei dati per potenzialmente migliorare le performance classificative.

La funzioni kernel sono generalmente delle funzioni non lineari di tipo polinomiale, radiali o con tangente iperbolica. L'applicazione di queste funzioni dovrebbe migliorare le performance classificatorie e l'overfitting dovrebbe essere contenuto nel caso in cui si utilizzino tanti dati; lo svantaggio principale sono il notevole aumento dei tempi computazionali.

5.4 Ridge Classifier

Il *Ridge Classifier* è la versione adattata alla classificazione della *Ridge Regression*, una variazione della normale regressione basata sui minimi quadrati. La particolarità della regressione è l'introduzione di una penalità, già incontrata nella *SVM*, per penalizzare i modelli complessi ed evitare l'overfitting. Nella regressione questa penalità è espressa aggiungendo alla formula dei minimi quadrati la somma dei coefficienti pesata per un valore variabile compreso nel range $[0, 1]$: più il valore è basso più la penalizzazione è lieve e viceversa; la scelta di questo parametro è molto importante in quanto si possono ottenere risultati molto diversi. Nel caso in cui il target non sia binario, il modello procede con una regressione ad output multipli, attribuendo la classe di appartenenza all'output più elevato.

Questo modello, nella sua forma lineare, è anche conosciuto come una SVM dei minimi quadrati. La funzione di costo nell'espressione (5.9) ha una formulazione molto simile a quella della SVM vista in precedenza, in questo caso la funzione è composta da due addendi: una parte di minimi quadrati ed una di penalizzazione:

$$J(\mathbf{w}) = \frac{1}{2m} \sum_{i=1}^{m} \Big(\varphi_{\mathbf{w}}(x_i) - y_i \Big)^2 + \frac{\alpha}{2m} \sum_{j=0}^{m} w_i^2 \tag{5.9}$$

Dove il parametro α a differenza della SVM è uguale a $\frac{1}{C}$. Nonostante la natura lineare, questo modello, la SVM e il modello esposto nel prossimo paragrafo, hanno delle buone performance classificatorie per i compiti di classificazione dei testi in quanto si lavora con un numero molto elevato di variabili e quindi lo spazio tende ad essere divisibile linearmente anche senza far ricorso a delle funzioni non lineari più complesse che rischiano di essere troppo specifiche portando all'overfitting. La gestione del vettore dei pesi e il parametro di penalizzazione, se scelto adeguatamente, permettono un ottimo compromesso fra precisione e generalibilità.

A differenza di altri modelli, l'utilizzo dei minimi quadrati può portare a dei vantaggi in termini computazionali in quanto la matrice di proiezione $(\mathbf{X^T X})^{-1}\mathbf{X^T}$ utilizzata per stimare i parametri della regressione, è calcolata una volta sola; più il numero di classi è elevato, più sono i vantaggi rispetto a modelli come la *Regressione Logistica*.

5.5 Regressione Logistica

La *Regressione Logistica* [49] è un modello lineare di classificazione che nella sua versione binaria si basa sulla funzione logistica (sigmoide) per la sua definizione: $g(x) = \frac{1}{1+e^{-x}}$. Nelle classificazioni con target non binario, la funzione sigmoide viene sostituita da una *softmax*, equazione (5.10), utilizzata anche in altri metodi di classificazione come le reti neurali:

$$softmax(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{5.10}$$

La funzione porta tutti i valori nel range $[0, 1]$ e la somma totale dei valori deve essere unitaria; in questo modo l'output è interpretabile come una probabilità e ad una osservazione viene attribuita la classe a cui corrisponde la probabilità più alta. La funzione di perdita è riportata nell'equazione (5.11):

$$J(\mathbf{w}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} 1(y^{(i)} = j)\log\left(\frac{\exp\theta_j^T x^{(i)}}{\sum_{l=1}^{k}\exp\theta_j^T x^{(i)}}\right) \tag{5.11}$$

Dove 1 rappresenta la funzione indicatrice e la quantità nel logaritmo non è altro che la generalizzazione dell'equazione (5.10). Anche in questa classe di modelli è presente la possibilità di introdurre un termine di penalizzazione per evitare l'overfitting.

5.6 Ricerca del minimo

La minimizzazione della funzione di perdita non è risolvibile linearmente, dunque metodi come l'algoritmo di *Newton-Raphson* sono utilizzati per la ricerca del minimo. Selezionato un punto di partenza x_0 l'algoritmo effettua lo sviluppo di Taylor al secondo ordine per trovare un nuovo punto x_1 nel suo intorno, diventando il

nuovo minimo. Lo sviluppo si ferma al secondo ordine in quanto è considerabile come l'equazione di una parabola dove per definizione il vertice è il punto di minimo della funzione. L'algoritmo procede per via iterativa fino a raggiungere un punto x_k considerabile come il minimo. L'algoritmo infatti non raggiungerà mai il reale valore minimo, ma fornirà un'approssimazione con un errore trascurabile tale da considerare quel punto come il minimo globale. L'algoritmo seleziona il nuovo punto ad ogni iterazione basandosi sulle derivate prime e seconde della funzione in quel punto e continua ad aggiornarsi fino a quando fra un'iterazione e la successiva l'aggiornamento non varia di una quantità variabile ϵ solitamente molto piccola. Uno dei limiti di questo metodo è l'individuazione del minimo globale: se la funzione da minimizzare ha una forma non canonica con molti minimi locali, è possibile che l'algoritmo converga ad un minimo locale e non globale.

Sia x^* il punto di minimo della funzione $f(x)$ e siano rispettivamente $f^{'}$ e $f^{''}$ la derivata prima e derivata seconda della funzione, con $f^{''}(x^*) > 0$. Sia x_k l'approssimazione di x^* identificata al k-esimo passaggio dall'algoritmo, lo sviluppo di Taylor di secondo ordine della funzione f intorno al punto x_k è rappresentata nell'equazione (5.12):

$$\min_p f(x_k + p) \approx f(x_k) + f^{'}(x_k)p + \frac{1}{2}f^{''}(x_k)p^2 \tag{5.12}$$

Ponendo la derivata a zero si ottiene la soluzione $p = \frac{f^{'}(x_k)}{f^{''}(x_k)}$, da cui l'aggiornamento dell'equazione (5.13):

$$x_{k+1} = x_k + p = x_k + \frac{f^{'}(x_k)}{f^{''}(x_k)} \tag{5.13}$$

Nel caso a più dimensioni l'aggiornamento, in equazione (5.14), ha una formulazione simile a quella del caso ad una dimensione:

$$x_{k+1} = x_k - [f^{''}(x_k)]^{-1} f^{'}(x_k) \tag{5.14}$$

Dove $f^{'}(x_k) = \nabla f(x_k)$ è il gradiente, ovvero il vettore delle derivate prime e $f^{''}(x_k) = \nabla^2 f(x_k) = H_f(x_k)$ è la matrice delle derivate seconde, detta matrice Hessiana. La principale problematica del metodo risiede nell'inversione della matrice Hessiana: l'inversione di una matrice non è un compito molto rapido da eseguire e la complessità di questa operazione è di $O(n^3)$ [50], i tempi computazionali crescono in modo più che esponenziale all'aumentare della dimensionalità. Per questa ragione i modelli di machine learning utilizzano degli approcci definiti *quasi Newton-Raphson* per ridurre la complessità dei calcoli, nello specifico, nelle seguenti sottosezioni, saranno trattate due differenti metodologie.

5.6.1 L-BFGS

Gli algoritmi *quasi Newton-Raphson* sono costruiti in modo tale da non ricorrere al calcolo dell'inversa della matrice Hessiana e di utilizzare una sua approssimazione. Sia H_k dunque l'approssimazione di $[f^{''}(x_k)]^{-1}$, l'aggiornamento dei nuovi punti è dato dall'equazione (5.15):

$$x_{k+1} = x_k + \alpha_k d_k \tag{5.15}$$

Dove $d_k = -H_k f^{'}(x_k)$ e α_k è un vettore positivo che minimizza la funzione f rispetto a d_k e che rispetta la condizione di Wolfe [51]. La matrice approssimata deve poi essere aggiornata ad ogni iterazione per ricavare H_{k+1}, applicando lo sviluppo di Taylor al secondo ordine come visto in precedenza, si ricava l'approssimazione espressa nell'equazione (5.16):

$$H_{k+1}\gamma_k = \mathbf{p}_k \tag{5.16}$$

Dove $\gamma_k = f^{'}(x_{k+1}) - f^{'}(x_k)$ e $\mathbf{p}_k = x_{k+1} - x_k$. Nonostante i due vettori siano espressi tenendo presente della derivata prima, è possibile dimostrare che entrambi sono un'approssimazione rispettivamente della matrice Hessiana e della sua inversa, in questa maniera si pongono le basi per un aggiornamento senza dover far ricorso alla matrice delle derivate seconde.

L'equazione (5.16) rappresenta un sistema lineare di n equazioni con $\frac{n(n-1)}{2}$ gradi di libertà. Esistono diversi modi per trovare le soluzioni di quella equazione, uno di questo è l'algoritmo **BFGS** che prende il nome dai suoi autori *Broyden–Fletcher–Goldfarb–Shanno* [52]. L'idea dell'algoritmo è di trovare una matrice $B_k \approx f''(x_k)$ tale per cui il suo aggiornamento B_{k+1} rispetti l'equazione (5.16) con i vettori posti nell'altro lato dell'equazione, $B_{k+1}\mathbf{p}_k = \gamma_k$ e il conseguente aggiornamento avviene come descritto nell'equazione (5.17):

$$B_{k+1} = B_k + a_k U + b_k V \tag{5.17}$$

Con a_k eb_k vettori di coefficienti e U, V due matrici simmetriche di rango unitario. Con queste condizioni si ha che $B_k H_k = I$, matrice di identità e lo stesso vale per tutti gli aggiornamenti ad ogni iterazione. Con queste approssimazioni si realizza l'algoritmo di Newton-Raphson senza dover invertire la matrice Hessiana, raggiungendo una complessità computazione di $O(n^2)$.

Questa soluzione può essere ulteriormente approssimata con un altro metodo denominato **L-BFGS** (*Limited-memory BFGS*), variazione dell'approssimazione precedente e implementata nei principali modelli di machine learning [53] [54]. Per velocizzare ulteriormente la ricerca del punto di minimo, questo algoritmo non tiene in memoria tutta la matrice di tutti gli step precedenti, ma considera solamente un numero di vettori $m \ll n$ di step precedenti, solitamente tra i $10-20$. Questi vettori sono costruiti in modo tale che siano una rappresentazione dell'intera matrice e che tengano conto delle informazioni sulla curvatura delle iterazioni più recenti. Questa soluzione prevede un vantaggio e anche uno svantaggio:

- La memorizzazione di un numero ristretto di vettori permette di avere più memoria libera e di operare con elementi più contenuti diminuendo i tempi computazionali ma mantenendo la logica originale dei metodi Newton-Raphson.
- L'algoritmo che si vuole semplificare è già se stesso un'approssimazione di un altro algoritmo, introducendo un altro livello di approssimazione si rischia di ottenere dei risultati significativamente peggiori.

5.6.2 SAG

Il secondo metodo che è stato implementato è denominato **SAG** (Stochastic Average Gradient) variazione del metodo *Stochastic Gradient Descent* che utilizza un'approssimazione stocastica della discesa del gradiente per la minimizzazione di una funzione che può essere vista come la somma finita di tanti elementi. La formulazione del problema generico è presentato nell'equazione (5.18):

$$Q(w) = \frac{1}{n} \sum_{i=1}^{n} Q_i(w) \tag{5.18}$$

con il parametro w da ricercare per la minimizzazione di $Q(w)$ e la quantità $Q_i(w)$ è interpretabile come la funzione di perdita della i-esima osservazione. Per minimizzare questa quantità un metodo di gradiente discendente viene applicato tale per cui l'algoritmo assume nella sua forma iterativa l'equazione (5.19) riportata di seguito:

$$w := w - \eta \nabla Q(w) = w - \eta \sum_{i=1}^{n} \frac{\nabla Q_i(w)}{n} \tag{5.19}$$

dove η è un iperparametro che è chiamato *learning rate* che definisce la variazione del passo ad ogni iterazione. Nella versione stocastica del gradiente discendente, la stima del minimo della funzione è ottenuta secondo i seguenti step:

- Si seleziona un vettore iniziale w e η.
- Si calcola $w := w - \eta \nabla Q_i(w)$ per $i = 1, \ldots, n$ e si aggiorna η ad ogni iterazione.
- Se non è raggiunta la convergenza dopo n iterazioni si rimescolano le osservazioni del dataset.
- I criteri di arresto sono gli stessi del metodo precedente e di tutti i metodi appartenenti alla classe dei metodi che implementano delle approssimazioni: la differenza tra un'iterazione e la successiva del valore di w è inferiore ad una soglia predefinita o l'aggiornamento di η è pressoché nullo.

Il metodo SAG [55] non è altro che il metodo presentato con una struttura dei pesi w differente: come suggerisce il nome, l'algoritmo tiene presente il valore dei pesi ad ogni iterazione e una volta raggiunta la convergenza il valore di w è sostituito con $\bar{w} = \frac{1}{t}\sum_{i=0}^{t-1} w_i$. Poiché questa soluzione tiene in memoria i valori ottenuti nelle precedenti iterazioni, converge più rapidamente di un normale metodo di gradiente discendente stocastico [56] [57] e per via della natura dell'algoritmo, risulta scalabile anche per dataset molto grandi con una complessità di $O(n)$ [58].

Capitolo 6

Sentiment Analysis

6.1 Libreria

Per la Sentiment Analysis è stata utilizzata la libreria Vader [59] specializzata nell'analisi di testi provenienti da social media e più in generale di UGC. Gli autori nel paper hanno provato la libreria confrontato diversi lexicon già esistenti e tecniche di machine learning ottenendo delle buone prestazioni. La libreria ha delle buone performance su una vasta gamma di testi, tra cui anche le recensioni di prodotti e film [60]. La libreria è allenata in modo da poter gestire non solo il testo che viene normalmente utilizzato ma anche le sfumature che sono presenti nei social [61]:

- Gestione delle negazioni, anche nelle loro forme contratte.
- Analisi delle parole scritte in maiuscolo.
- Parole che amplificano il sentiment e parole che lo minimizzano.
- Gestione degli slang
- Gestione degli slang come amplificatori.
- Emoticons sia testuati che in formato UTF-8

Nelle analisi di pre-processing sono state applicate alcune funzioni che hanno alterato il testo iniziale ma non hanno compromesso il corretto funzionamento della libreria. Per la gestione delle negazioni, tutte le forme contratte sono state riportate nella forma estesa in quanto era necessario per il tagging, le maiuscole e minuscole

sono state tenute nelle diverse fasi di analisi principalmente per poter identificare correttamente i nomi delle personalità o i titoli dei film. Il mantenere le parole nel loro formato originale non tutto in minuscolo ha aiutato il corretto utilizzo di una funzionalità della libreria, e allo stesso tempo i termini sono stati opportunamente portati in minuscolo temporaneamente nei diversi step precedenti, ad esempio per il clustering, per evitare di trovare termini duplicati vista la natura case sensitive del linguaggio di programmazione utilizzato. Gli slang potrebbero essere stati modificati con la spelling correction, proprio per questo si valuterà alla fine la differenza nei risultati. Le emoticons sono state effettivamente modificate, così come tutto il resto della punteggiatura, ma questo era necessario per identificare i pattern e inoltre non dovrebbero rappresentare degli elementi con alta frequenza all'interno delle recensioni non trattandosi di dati provenienti da un social media come Twitter o Facebook.

La libreria fa riferimento ad un vocabolario in formato testuale al cui interno sono presenti le parole riconosciute, un valore medio di sentiment che varia nel range $[-4, 4]$ e un valore di standard deviation in base a quanto il sentiment stimato di quella parola varia nei testi raccolti per costruire la libreria. La libreria offre quattro diversi elementi di sentiment in input inserita una parola o una frase, questi valori sono: la polarità negativa, la polarità positiva, la neutralità e un valore chiamato *compound* che è un unico valore, compreso fra $[-1, 1]$ che racchiude le informazioni dei tre elementi precedenti. Il valore di questo elemento è calcolato come riportato nell'equazione (6.1) [62]:

$$compound = \frac{score}{\sqrt{score^2 + \alpha}} \tag{6.1}$$

Dove lo *score* è il valore normalizzato di sentiment di una frase e α un valore di smoothing che di default è selezionato a 15. In breve questo valore di compound, che sarà utilizzato in tutte le future analisi, prende come riferimento uno score che è calcolato come una normalizzazione dello score di una frase che è calcolata a sua volta dalla somma dei sentiment presenti nel vocabolario tenendo presente la loro deviazione standard.

La funzione è stata calcolata su tutti i 14 milioni di pattern rilevanti identificati, ricavando un valore di compound. Prima di applicare la funzione di sentiment, la

frase è stata modificata in modo tale da rimuovere il titolo dell'opera di riferimento dal testo. Anche se si sono presi i pattern, non è da escludere che alcuni di essi potessero contenere il titolo dell'opera completo o in parte che, come già menzionato in precedenza, è doveroso rimuovere in quanto potrebbe influenzare il sentiment globale. Questa operazione è stata effettuata estraendo il testo del titolo, rimuovendo l'anno di realizzazione dove presente, si è considerato il titolo completo e confrontato con il contenuto del pattern: in caso affermativo quella specifica sequenza di parole è stata rimossa. Per i possibili titoli spezzati fra due pattern, si è confrontata la presenza di una porzione del titolo in un pattern e in caso affermativo è stato realizzato un confronto con il pattern successivo.

6.2 Definizione di un nuovo vocabolario

Un problema frequente nella Sentiment Analysis è capire se il lexicon di riferimento è adatto al progetto su cui viene applicato. Nella sezione precedente, quando è stata introdotta la libreria Vader, si è parlato della struttura del vocabolario interno, che è composto da uno score e un valore di deviazione standard poiché il valore di sentiment che assume una parola non è universale ma può avere diverse sfumature a seconda dell'argomento trattato. Per questo motivo in diverse situazioni è necessario attribuire un nuovo sentiment ad alcune parole per avere una rappresentazione migliore all'interno del contesto.

Nel lavoro in questione si è notato che diverse parole, specialmente con sentiment di base negativo, potevano essere catalogate in maniera differente. Parole come "*killer*", "*war*" hanno un sentiment negativo molto accentuato con valori di compound intorno a -0.50 ma che spesso non trovano riscontro con quello che è l'argomento del testo. Per esempio se una di queste parole è inserita all'interno di un pattern in cui si parla della trama di un film, il sentiment di queste parole dovrebbe essere nullo o certamente non negativo. Questo comportamento è stato notato sia nella libreria Vader, sia in altre librerie, fra cui anche quella di WordNet.

Per risolvere questo problema, molto più frequente per i termini negativi, sono state prese le recensioni con uno score molto positivo di almeno 8/10 stelle e sono stati salvati tutti i termini al loro interno nel caso in cui il sentiment calcolato con

la libreria fosse stato negativo. La logica dietro a questa scelta è di individuare quei pattern considerati come negativi ma che difficilmente dovrebbero essere presenti in quelle recensioni poiché il giudizio è molto positivo. Altri lavori in letteratura hanno presentato dei metodi differenti per la creazione di un proprio vocabolario, come l'*information gain* basandosi sulla probabilità di appartenenza di una parola ad una determinata classe. In questo lavoro non è stato seguito questo approccio in quanto non si hanno i reali valori dei label ma sono stati ricavati in base ai termini utilizzati nella cluster analysis e si è voluto sfruttare un'altra informazione presente, tenendo conto che non tutti gli utenti giudicano allo stesso modo e che due persone possono gradire allo stesso modo un film ma giudicarlo con due punteggi differenti.

Una volta trovate le parole si è effettuato un conteggio per vedere il rapporto tra la frequenza di questi termini negativi nelle recensioni positive e gli stessi termini nelle recensioni negative. Filtrando le parole con poche occorrenze e adattando la numerosità delle recensioni nei due insiemi si è trovato un elenco di termini fortemente negativi ma con un utilizzo non significativamente differente fra i due insiemi. La libreria Vader permette di modificare il vocabolario senza passare per il file di base ma direttamente con una delle sue funzioni, il sentiment di queste parole è stato portato vicino allo zero. Gli stessi passaggi sono stati fatti per individuare le parole positive di default ma da non considerare tali in questo lavoro; in questo caso il numero di candidati era molto inferiore.

6.3 Sentiment Abruptness

Come step aggiuntivo sarà implementata la tecnica di *Sentiment Abruptness* introdotta nel paper di *Patra et al.* (2008) [22], utilizzata per una semplice sentiment analysis e non aspect-based. L'idea è quella di creare un grafico su un piano cartesiano dove sull'asse delle ascisse è rappresentata la posizione delle parole nella frase e sull'asse delle ordinate il corrispondente valore di sentiment. Passate tutte le parole si crea una spezzata che oscilla in positivo e in negativo a seconda delle parole incontrate e secondo gli autori se si trova un cambiamento molto rapido nel sentiment, quindi una rapida successione di parole negative e positive o viceversa è molto probabile che si tratti di ironia o sarcasmo.

Questo cambiamento repentino di sentiment si può calcolare tramite le coordinate dei punti e le curvature di Menger. Sia P_i un generico punto sul piano cartesiano di coordinate (x_i, y_i) dove x_i rappresenta la sua posizione all'interno della frase/pattern e y_i il suo valore di sentiment il valore di *Sentiment Abruptness* per una tripla di punti è calcolato come segue nell'equazione (6.2):

$$\mathbf{K} = \frac{4 \times f(P_1, P_2, P_3)}{\sqrt{g(P_1, P_2, P_3)}} \tag{6.2}$$

$$f(P_1, P_2, P_3) = \frac{1}{2}\Big|(x_2 - x_1)(y_3 - y_1) - (x_3 - x_1)(y_2 - y_1)\Big| \tag{6.3}$$

$$\begin{aligned} g(P_1, P_2, P_3) = ((x_2 - x_1)^2 + (y_2 - y_1)^2)((x_3 - x_1)^2 \\ + (y_3 - y_1)^2)((x_3 - x_2)^2 + (y_3 - y_2)^2) \end{aligned} \tag{6.4}$$

Le equazioni (6.3) e (6.4) sono formule utilizzate per le curvature di Menger e rappresentano il rapporto tra l'area del triangolo e la lunghezza dei suoi lati. Il valore viene calcolato per ogni tripla di punti e, successivamente sommato termine per termine; alla fine del pattern il valore viene diviso per il numero di termini considerati in modo da normalizzare i valori e non premiare i pattern più lunghi. Per i valori di sentiment sono stati presi i valori non normalizzati presenti nella libreria Vader con un range di $[-4, 4]$ lo stesso range utilizzato nel paper.

Tutti i pattern che hanno ottenuto un valore di *Sentiment Abruptness* maggiore di 0.10, il loro sentiment è stato moltiplicato per -1 per seguire quello che è il concetto di ironia e sarcasmo, un sentiment opposto a quello che è letteralmente presente. Il valore scelto per definire un pattern ironico o meno è stato scelto in base ai risultati ottenuti nel paper, in cui questo valore garantisce di individuare un numero ristretto di pattern non stravolgendo il sentiment di molte osservazioni rischiando di colpire anche pattern che non sono ironici.

Completata questa operazione si sono assegnate le etichette relative al sentiment ad ogni pattern. Il valore di cut off per separare le classi è quello usato dalla libreria Vader e globalmente quello più utilizzato: valori maggiori di 0.05 l'etichetta risultante è positiva, valori compresi tra $[-0.05, 0.05]$ etichetta neutra e i rimanenti

negativi. Il numero di pattern catalogati come negativi è la metà di quelli neutri e positivi che sono sulle stesse cifre. Questa differenza è dovuta a due fattori: il numero di recensioni positive, con valutazione sopra le cinque stelle è maggiore rispetto a quelle negative e la creazione del vocabolario per questo lavoro ha abbassato il sentiment di più parole negative che positive.

Capitolo 7

Classificazione

La classificazione è stata effettuata utilizzando i modelli elencati nel *Capitolo 5* su quattro diversi dataset per realizzare dei confronti e valutando il miglioramento o il peggioramento delle performance classificatorie relativamente alla spelling correction e all'individuazione del sarcasmo. Di conseguenza i quattro dataset citati saranno: uno senza spelling correction né identificazione del sarcasmo, uno con entrambi e gli altri due con una delle due tecniche. Per poter rendere comparabili i risultati dei dataset si è deciso di prendere i risultati della cluster analysis in modo tale che sia nel dataset con spelling correction che senza si avessero lo stesso numero di categorie. Le categorie individuate dalla cluster analysis sono state 12 a cui se ne sono aggiunte altre due:

- Una classe relativa ai pattern che non presentavano nessuna parola o sinonimo individuato dalla cluster analysis. Non potendo avere delle informazioni sulle etichette da assegnare, l'unica soluzione è stata quella di generare una nuova classe di pattern al cui interno non compaiono parole ad alta frequenza.

- Si è creata una classe apposita per i pattern al cui interno compariva il nome di una personalità legata al film. Nel caso in cui si fosse riscontrato il nome e il cognome di una personalità, il pattern è stato catalogato in una categoria a parte, indipendentemente dal numero delle altre parole individuate dalla cluster analysis.

Ai 14 tipi differenti di pattern identificati si è aggiunta l'informazione relativa al sentiment, positivo, negativo o neutro. Il compito di classificazione ha quindi visto l'assegnazione di una classe per ogni pattern per le 42 possibili alternative. Il nome

delle classi prende il nome dal termine più frequente all'interno delle parole in un cluster o in alcuni casi è un termine che riassume quelli più comuni. Le categorie che sono state identificate sono:

- Emozioni: insieme di emozioni che possono essere riferite a come una persona ha ricevuto il film e dalle sensazioni provate durante la visione.
- Intrattenimento: una classe simile a quella precedente ma con sfumature differenti.
- Generico: un cluster generico di parole.
- Trama: pattern incentrati sulla trama.
- Sviluppo: legato alla classe precedente, indica i pattern che si concentrano su come il film procede sotto diversi aspetti.
- Generico 2: un altro cluster di parole generiche.
- Attori: commenti sugli attori protagonisti.
- Generico 3: un terzo cluster di parole generiche.
- Musica: pattern sulla colonna sonora e gli effetti sonori.
- Produzione: come è stato gestito il film a livello di produzione, sono spesso commenti riferiti ai registi.
- Globale: una valutazione globale del film e non specifica su un aspetto, può essere interpretato come un giudizio generale.
- Scena: pattern che si soffermano nell'analizzare una scena specifica.

La cluster analysis non ha prodotto dei risultati eccellenti e quindi è normale che ci siano dei cluster simili fra loro e poco differenziati. I cluster definiti "generici" sono quei cluster al cui interno era presente un numero molto limitato di parole che non era possibile identificare univocamente con un termine che le rappresentasse tutte. Nonostante i pattern riferiti a queste classi sono solitamente limitati, l'introduzione di queste tre classi separate porta a dei risultati di classificazione migliori rispetto a non considerarli o a creare un'unica classe generica.

Le analisi sono state realizzate in parallelo in locale su una macchina con CPU Intel Core i7-8700K e su Google Colab [63], una piattaforma per realizzare codice in Cloud con la stessa struttura di Jupyter Notebook. I tempi di esecuzione non hanno riscontrato differenze significative tra i due notebook eccetto per la SVM, i cui dettagli saranno riportati in seguito.

I cinque modelli presentati nel Capitolo 5 sono stati selezionati fra un insieme più ampio in base alle performance classificatorie su una porzione del dataset e ai tempi computazionali, con l'eccezione dei metodi Naive Bayes che sono stati inseriti in quanto solitamente utilizzati come baseline.

7.1 Classificazioni separate

7.1.1 Classificazione Label

Per avere un'analisi più ampia prima di procedere con la classificazione delle 42 categorie sono stati realizzati dei modelli per la classificazione dei 14 tipi di pattern e del loro sentiment separatamente. I risultati per i pattern sul dataset con spelling correction e senza sono riportati nella *Tabella 7.1*:

Modello	Accuratezza	Tempo di esecuzione	Accuratezza spelling correction	Tempo di esecuzione spelling correction
SVM	0.723	11m 3s	0.721	10m 37s
LogReg (sag)	0.721	5m 20s	0.719	4m 55s
Ridge (auto)	0.714	7m 4s	0.713	7m 7s
Ridge (sag)	0.714	32m 22s	0.713	33m 58s
LogReg (lbfgs)	0.709	15m 8s	0.708	15m 23s
Complement NB	0.679	12.7s	0.679	12.8s
Naive Bayes	0.657	12.9s	0.659	13.1s

Tabella 7.1: Classificazione dei pattern con e senza spelling correction

L'ordine dei modelli per accuratezza è la stessa sia nel dataset originale sia su quello dove è stata realizzata la spelling correction. I valori di accuratezza sono pressoché identici, con un leggero vantaggio nel dataset senza la spelling correction. La Support Vector Machine è il modello migliore raggiungendo un'accuratezza di 0.723 e 0.721, seguita dalla Regressione Logistica utilizzando il metodo del gradiente discendente. Lo stesso modello con la minimizzazione della funzione di perdita ottenuta con un metodo quasi Newton-Raphson ha performance peggiori ed è anche più dispendioso a livello computazionale. Una delle possibili spiegazioni può essere l'eccessivo numero di approssimazioni che applica questo metodo, come esposto nel Capitolo 5 e di conseguenza ne risentono le performance classificative.

Il classificatore Ridge di default utilizza un metodo automatico per la minimizzazione della funzione di perdita e non si notano differenze con altri metodi, tra i quali quello del gradiente discendente applicabile anche in questo caso. Le performance sono indifferenti, varia solo il tempo di esecuzione che è più di quattro volte tanto con il metodo del gradiente discendente. Questo metodo dovrebbe essere efficace per i dataset con un numero elevato di osservazioni ma, mentre per il modello logistico si è registrato questo miglioramento con tempi di esecuzione tre volte inferiori, per questo classificatore l'algoritmo richiede un maggior numero di iterazioni che si traduce in tempi di esecuzioni molto più lunghi.

I classificatori Naive Bayes sono come prevedibile i modelli peggiori anche se non c'è una differenza nettissima dagli altri modelli. Come già anticipato in precedenza il modello Complement Naive Bayes ha delle performance leggermente migliori.

Per queste classificazioni e tutte quelle successive la divisione tra train e test set è stata realizzata con le proporzioni di default con il 70% delle osservazioni nel train e il 30% nel test. La suddivisione delle 3537428 osservazioni all'interno delle classi nel test set era molto sbilanciata, passando da massimo di 654982 ad un minimo di 9569. Nonostante queste differenze molto evidenti, non si è verificata una grossa differenza di peformance classificatorie nelle singole classi; eventuali metodi di ricampionamento saranno presi in considerazione successivamente nella classificazione congiunta di pattern e sentiment.

7.1.2 Classificazione Sentiment

La classificazione del sentiment è stata realizzata, anche in questo caso, su due dataset: uno senza l'identificazione del sarcasmo e una con. Alcuni dei modelli utilizzati nella classificazione dei pattern non sono stati replicati in questa situazione sia perché le performance non erano al pari di modelli simili ma con parametri diversi, sia perché molto più lenti. I risultati delle due classificazioni sono riportati di seguito nella *Tabella 7.2*:

Modello	Accuratezza	Tempo di esecuzione	Accuratezza sarcasm detection	Tempo di esecuzione sarcasm detection
LogReg (sag)	0.972	3m 29s	0.963	3m 53s
SVM	0.967	2m 21s	0.963	2m
Ridge (auto)	0.959	3m 16s	0.948	1m 59s
Naive Bayes	0.938	9.2s	0.913	9.5s
Complement NB	0.922	9.2s	0.911	9.4s

Tabella 7.2: Classificazione dei pattern con e senza identificazione del sarcasmo

Questa classificazione è semplificata dal fatto che sono prese in considerazione solamente tre classi, positivo, negativo e neutro, ma i risultati sono molto buoni con i modelli migliori che sfiorano la perfezione. In questo caso primo modello per accuratezza è la Regressione Logistica, che è stata provata solamente con il metodo del gradiente discendente. La Support Vector Machine è seconda e al pari nel dataset in cui è stata realizzata un'identificazione del sarcasmo. Il classificatore Ridge è a metà con un distacco in proporzione significativo tra i due modelli migliori e i metodi di Naive Bayes. Per questi ultimi è interessante notare che il metodo di base ottiene delle performance migliori a differenza della classificazione precedente.

Anche in questo caso la numerosità all'interno delle classi non era equa, nello specifico le osservazioni con etichetta negativa erano la metà di quelle positive e neutre che avevano sostanzialmente lo stesso numero. Questa disparità si spiega principalmente per il fatto che la maggior parte delle recensioni era positiva, un dato pronosticabile in quanto sono stati presi i film più famosi che generalmente hanno valutazioni alte se non altissime e i primissimi rappresentano una buona parte delle osservazioni. Una seconda spiegazione può derivare dal fatto che nella

costruzione del vocabolario specifico per questo lavoro, diverse parole negative sono state considerate con sentiment nullo, mentre queste modifiche erano molto più ristrette per i termini positivi.

Preso atto di questi risultati, nella classificazione completa con 42 categorie ci si potrebbe aspettare un'accuratezza simile a quella ottenuta con i modelli addestrati sulle 14 tipologie di pattern poiché il sentiment è individuato correttamente nella grande maggioranza delle osservazioni e non dovrebbe far crollare le performance di classificazione..

7.2 Classificazione mista

Per la classificazione finale sono stati provati diversi modelli con diverse combinazioni di parametri e diversa generazione delle features con il metodo TF-IDF. I parametri per la creazione delle features sono i seguenti:

- Numero di features: massimo 35000. Nella classificazione non sono stati considerati i termini meno frequenti in quanto più informazioni portavano ad un leggero peggioramento delle performance di classificazione. Tra i diversi valori provati i risultati erano costanti nel range $20000 - 50000$ al di fuori di esso l'accuratezza iniziava a scendere anche in maniera significativa.

- Utilizzo dei bigrammi. La classificazione tenendo in considerazione le parole singole non portava a dei buoni risultati, mentre le performance erano notevolmente migliorate considerando congiuntamente le singole parole e i bigrammi. Si è provato ad andare anche oltre considerando assieme a unigrammi e bigrammi anche i trigrammi. La classificazione aveva gli stessi risultati, aggiustando in modo appropriato il numero di features da considerare, ma i tempi di computazione aumentavano.

- La libreria *scikit-learn* permette di utilizzare un filtro di stopwords che è differente da quello già utilizzato nella libreria *nltk*. La combinazione di questi due insiemi di stopwords portava ad eliminare delle parole che però risultavano utili per la classificazione abbassando le performance classificative.

La divisione tra il train e test set è stata realizzata in modo stratificato, ovvero mantenendo le proporzioni delle classi fisse in entrambi gli insiemi e con un seme fisso per replicare l'identica divisione delle osservazioni per tutti i modelli. Le analisi sono state effettuate su dataset con e senza spelling correction e identificazione del sarcasmo nelle quattro combinazioni possibili; i risultati sono elencati in *Tabella 7.3*:

Modello	Accuratezza No SC, No S	Accuratezza No SC, S	Accuratezza SC, No S	Accuratezza SC, S	Tempo medio di esecuzione
LogReg (sag)	0.687	0.697	0.688	0.687	20m 4s
SVM	0.670	0.672	0.664	0.661	28m 50s
Naive Bayes	0.602	0.624	0.605	0.600	7m 4s
Ridge (auto)	0.548	0.588	0.543	0.544	30m 46s
Complement NB	0.504	0.546	0.505	0.508	6m 58s

Tabella 7.3: Classificazione di pattern e sentiment

Le sigle *SC* e *S* indicano rispettivamente *spelling correction* e *sarcasm*. Rispetto alle due classificazioni precedenti ci sono delle piccole variazioni. I tempi di esecuzioni sono aumentati per tutti i modelli, principalmente per i Naive Bayes che dovendo lavorare con più categorie hanno visto aumentare il numero di operazioni per il calcolo delle probabilità con un tempo totale aumentato di oltre 20 volte. Per gli altri modelli il tempo è aumentato ma in maniera più contenuta.

Il modello logistico si riconferma il modello migliore sia in termini di accuratezza sia in termini di velocità. Un riassunto della matrice di errata classificazione è presentata di seguito in *Figura 7.1* relativamente alla classificazione del dataset senza spelling correction e con identificazione del sarcasmo, risultata la combinazione migliore in assoluto.

Le etichette sono ordinate per classi di pattern a cui è associato il relativo sentiment. La prima riga e colonna fanno quindi riferimento alla prima etichetta di pattern con il sentiment negativo, la seconda alla prima etichetta con sentiment neutro, la terza alla prima etichetta e con sentiment positivo. I valori sulla diagonale principale sono ovviamente in colore più scuro in quanto sono le osservazioni predette correttamente e fuori da essa si possono identificare dei pattern per le osservazioni classificate erroneamente. La maggior parte delle celle di colore più scuro segue un pattern diagonale, se attaccate alla diagonale principale indicano che la

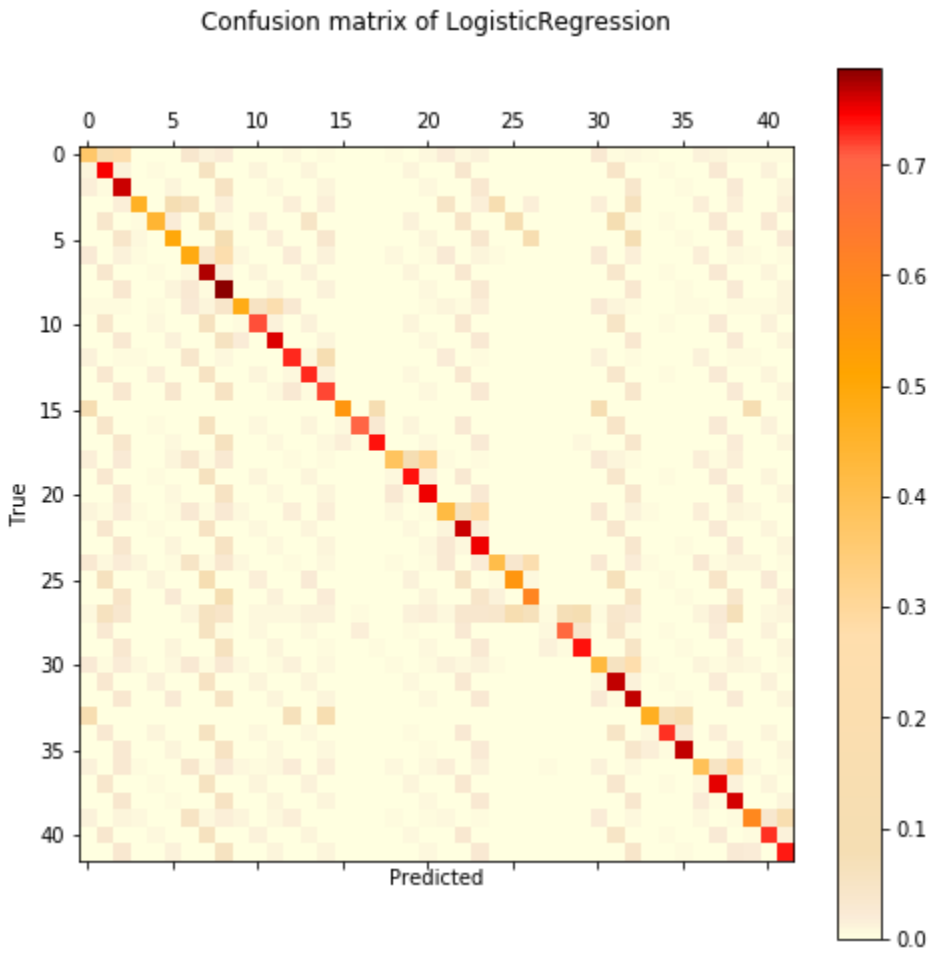

Figura 7.1: Matrice di classificazione del modello Logistico

classe del pattern è stata identificata correttamente ma il sentiment è errato. Negli altri casi si nota una certa uniformità che rispecchia quello ottenuto precedentemente nelle classificazioni separate ovvero che il sentiment è classificato correttamente ma la classe del pattern è errata.

I livelli di precisione per le singole classi non sono elevatissimi, si raggiunge il massimo a 0.80 per la classe dei giudizi positivi sulla colonna sonora e gli effetti sonori, mentre il minimo di 0.37 è per la classe delle emozioni negative sul film. Non ci sono errori sistematici di classificazione errata come si può notare anche dalla matrice, il che è un punto a favore del modello in quanto anche con poche osservazioni in una classe, il modello è in grado di riconoscerle, ma non c'è una classe per cui il modello non compie quasi nessun errore. Per gli altri modelli la matrice

ha una struttura simile e solo per i metodi di Naive Bayes si notavano delle righe completamente bianche a significare che delle classi erano completamente ignorate dal modello.

Alcuni esempi di pattern classificati sono riportati nella *Tabella 7.4*. I primi cinque esempi sono classificati correttamente sia a livello di pattern sia di sentiment. La classe è strutturata in modo tale da avere come primo numero la classe del pattern, il cui ordine segue l'elenco presentato all'inizio del capitolo e il secondo numero è il riferimento al sentiment. I numeri $4, 6, 8, 10$ sono relativi rispettivamente alla trama, agli attori, alla colonna sonora e al film in generale. Da notare come nel quinto esempio sia presente la parola "*unforseen*" che è uno dei termini a cui è stato cambiato il sentiment da negativo a neutro/positivo per via della sua valenza in questo ambito. Probabilmente senza questo accorgimento il pattern sarebbe stato classificato con un sentiment neutro o negativo, nonostante la presenza del termine "*nice*". Il sesto ed ultimo è invence un esempio di pattern classificato erroneamente: la classe reale è la trama, ma il modello lo classifica come un pattern generico.

Pattern	Classe reale	Classe predetta
fairy tale moments really good	4_1	4_1
bad every character	6_-1	6_-1
simple yet thoughtful soundtrack	8_1	8_1
movie made feel bored	10_-1	10_-1
nice unforseen twist	10_1	10_1
bubonic plague epidemic swept eastern later western world	4_0	5_0

Tabella 7.4: Esempi di classificazione di pattern

Per gli altri modelli la Support Vector Machine, che aveva ottenuto i risultati migliori nella classificazione del sentiment, si attesta leggermente indietro con un distacco di circa 0.02 in termini di accuratezza da parte dei modelli migliori. I tempi di esecuzione di questa tecnica sono interessanti in quanto è l'unico modello per cui c'è stata una notevole differenza fra il locale e Google Colab. In locale i modelli erano estremamente lenti con anche più di un'ora richiesta per l'esecuzione,

mentre su Google Colab raramente si sorpassavano i trenta minuti. Google Colab permette di utilizzare un acceleratore di GPU per velocizzare alcune operazioni e anche se, stando alla documentazione ufficiale di *sklearn* [64], la libreria non utilizza la GPU ma la CPU per le operazioni, per questo modello i tempi sono stati più che dimezzati.

I modelli di Naive Bayes hanno delle scarse performance e si aumenta ulteriormente la forbice presente fra il metodo di base e quello con l'utilizzo delle classi complementari. Il classificatore Ridge è il modello che soffre maggiormente l'aumento del numero di categorie passando ad essere il peggiore tra tutti, contrariamente a quello che dovrebbe essere il suo comportamento: all'aumentare del numero di classi dovrebbe scalare meglio di altri metodi come la Regressione Logistica.

Anche a livello di dataset ci sono delle differenze sostanziali rispetto a quello visto nelle classificazioni separate. Se prima le classificazioni sul dataset senza spelling correction e senza identificazione del sarcasmo erano le migliori, i modelli nel dataset *No SC, No S* sono al pari di quelle nei dataset *SC, No S* e *SC, S*. La Regressione Logistica ottiene un 0.01 di accuratezza in più nel dataset *No SC, S*, ovvero nel dataset con solamente l'identificazione del sarcasmo. Sono comunque dei valori talmente ravvicinati fra di loro che non si può parlare di un miglioramento significativo e che può essere soggetto alla divisione tra train e test set.

Un'importante questione ruota attorno al numero di osservazioni contenute all'interno delle classi: se già per la classificazione dei pattern c'erano delle classi molto sbilanciate, unendo le etichette con i sentiment questo problema è ulteriormente accentuato. Nel seguente paragrafo saranno applicate delle tecniche di ricampionamento per risolvere questo problema.

7.3 Ricampionamento

Analizzando i risultati dei modelli di classificazione può essere interessante effettuare un ricampionamento su alcune classi specifiche per valutare se è possibile migliorare le performance classificative. Il ricampionamento è stato realizzato utilizzando il metodo **SMOTE** (*Synthetic Minority Over-sampling TEchnique*). Differentemen-

te da un ricampionamento casuale che può creare problemi di overfitting, questo metodo genera delle nuove osservazioni come rappresentato in *Figura 7.2* [65].

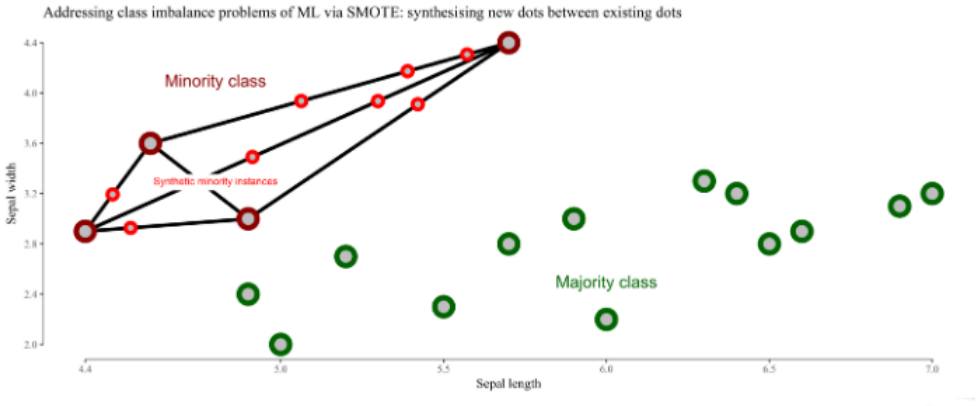

Figura 7.2: Metodo SMOTE per il ricampionamento

L'immagine fa riferimento al dataset *Iris* [66] molto famoso nella letteratura statistica. Come suggerisce il nome la tecnica non prende delle osservazioni già presenti nel dataset ma crea dei nuovi campioni sintetici. L'idea di base è quella di prendere un'osservazione nelle classi meno numerose e valutare un numero variabile di suoi vicini appartenenti alla sua stessa classe. Il metodo crea un nuovo campione con nuovi valori che si posizionano su delle linee immaginarie che collegano i punti, creando una sorta di campione medio. In questo modo sono create delle nuove osservazioni che hanno caratteristiche simili alla media complessiva delle osservazioni all'interno della classe e questo potrebbe migliorare le performance classificative.

Le analisi sono state realizzate solamente con il modello di Regressione Logistica andando a ricampionare le classi meno numerose e quelle che nonostante un elevato numero di osservazioni avevano delle scarse performance come le classi con il nome degli attori. La scelta di considerare un unico modello è stata presa sia tenendo presente i valori di accuratezza ottenuti in precedenza sia considerando i tempi elevati per creare questi nuovi campioni che dipendono fortemente dall'individuazione dei vicini tramite il metodo dei vicini più vicini. I risultati sono rappresentati nella *Tabella 7.5* dove nell'ultima colonna sono riportate le differenze in termini di accuratezza fra questi nuovi modelli e i corrispondenti nei rispettivi dataset:

Anche con un ricampionamento le performance restano invariate e l'unica variazione si può notare solamente al terzo decimale, il modello migliore rimane quello nel dataset senza spelling correction e individuazione del sarcasmo. Una delle possibili

Dataset	Accuratezza	Variazione
No SC, No S	0.689	+0.002
No SC, S	0.696	-0.001
SC, No S	0.685	-0.003
SC, S	0.687	-0.003

Tabella 7.5: Classificazione di pattern e sentiment

spiegazioni del perché neanche con il ricampionamento si ottengono dei miglioramenti può essere legato alla classificazione di base senza ricampionamento. Come si era notato dalla matrice di classificazione e dai valori di precisione, non si evidenziavano delle classi con delle criticità particolari, ma la classificazione sbagliava in maniera contenuta in modo abbastanza uniforme. Il problema e i limiti dei modelli sono da ricercarsi nelle caratteristiche macro e sistemare dei piccoli dettagli come possono essere le classi meno numerose non porta a dei miglioramenti evidenti.

Capitolo 8

Conclusioni

In questo lavoro si è creata un'architettura software di Aspect-based Sentiment Analysis in grado di analizzare il sentiment di diversi aspetti trattati in un ambito testuale generalizzabile a più contesti. La generalizzazione è uno dei punti di forza di questa architettura che può essere applicata a qualsiasi altro dataset testuale senza dover intervenire manualmente se non per piccoli dettagli come la denominazione dei cluster e il vocabolario specifico al contesto.

Uno dei principali scopi era quello di valutare la differenza e l'importanza dell'utilizzo di due tecniche relativamente conosciute all'interno del Natural Language Processing ma scarsamente utilizzate in contesti di Aspect-based Sentiment Analysis, la spelling correction e l'identificazione del sarcasmo.

Per la spelling correction il suo utilizzo è limitato in questo tipo di lavori in quanto la sua precisione solitamente non supera il 70% e quindi si procede con una certa cautela nel suo utilizzo. L'identificazione del sarcasmo è una delle tecniche più innovative che non è mai stata provata in un lavoro del genere ma è stata limitata a lavori di Sentiment Analysis. L'identificazione di questi commenti è molto importante in quanto portano con sé un giudizio opposto a quello testuale, ma come gli stessi autori della tecnica hanno scritto, alcune volte il sarcasmo non è colto dalle persone stesse ed è impensabile che una macchina ci riesca senza mai fallire. La ricerca su questo ambito è ancora molto indietro rispetto ad altre tecniche e le aspettative su questi metodi non può essere elevatissima.

Per i modelli di classificazione la Support Vector Machine è generalmente considerato come il miglior modello per la classificazione dei testi e benché abbia ottenuto prestazioni importanti non è riuscita a raggiungere la Regressione Logistica nono-

stante siano state valutate più configurazioni di parametri. Bisogna sottolineare che non sono stati provati dei kernel non lineari in quanto i tempi di elaborazione aumentavano in modo esponenziale e non erano conciliabili con quattro analisi distinte; inoltre la soluzione che si è dimostrata più veloce, Google Colab con l'acceleratore di GPU non permette di far girare un codice per più di 12 ore consecutive.

I valori di accuratezza sfiorano lo 0.70 che è generalmente considerato una baseline per distinguere i buoni modelli da quelli scadenti e questi risultati si allontanano dal miglior risultato assoluto nella letteratura per la Aspect-based Sentiment Analysis che è di 0.85. Questo lavoro però ha avuto delle differenze sostanziali da tutti gli altri in letteratura:

- Il numero delle osservazioni era notevolmente superiore, oltre 700 mila commenti, contro i 5-10 mila che sono stati utilizzati negli altri lavori. Questo numero contenuto di osservazioni ha permesso agli autori di classificare manualmente una osservazione oppure un'etichetta era già disponibile nel dataset utilizzato. In questo progetto le etichette non erano note e sono state create utilizzando una cluster analysis che non può avere una precisione assoluta.

- Il numero di classi considerato, 42, è notevolmente superiore a quello degli altri studi che si fermavano a circa 15. Come evidenziato in precedenza un maggior numero di classi non significa necessariamente una peggior classificazione, ma certamente il lavoro è molto più complicato e in combinazione con il punto precedente, gli elevati tempi di computazione non hanno permesso di esplorare alcune combinazioni di parametri o altre tecniche che avrebbero aiutato seppur in modo contenuto.

L'aspetto buono della classificazione è che per tutte le classi il modello migliore era in grado di classificare correttamente la maggior parte delle osservazioni con solamente le classi relative alle emozioni, ai pattern generali con termini non contenuti nella cluster analysis e quella con i nomi delle personalità ad avere dei valori di precisione inferiore a 0.70 per tutte le tre classi di sentiment. Le criticità principali erano poi relative alle classi negative in quanto le recensioni con voti bassi erano in numero molto minore rispetto a quelle positive e neutre, me neanche con le tecniche di ricampionamento è stato possibile migliorare le performance.

I valori di accuratezza nei quattro dataset sono talmente simili fra di loro che non si può parlare di una differenza significativa ma potrebbe cambiare in base anche

solamente al seme utilizzato. La spelling correction non sembrerebbe portare a dei risultati migliori mentre l'individuazione del sarcasmo, specialmente se affiancata a dei pattern senza spelling correction, ottiene dei risultati interessanti. Se si volesse azzardare una spiegazione di questo comportamento si potrebbe ipotizzare che la spelling correction elimina tutte le sfumature di slang o molto informali che potrebbero essere accostate a dei commenti sarcastici e l'identificazione di questo numero ristretto di pattern particolari potrebbe essere utile in termini di classificazione.

Dei possibili miglioramenti a questo lavoro possono essere la considerazione degli aspetti impliciti ricreandoli basandosi su delle parole chiave e confrontando la frequenza delle parole antecedenti e successive. Dato che la cluster analysis gioca un ruolo importantissimo all'interno della classificazione definendo i tag dei pattern, provare delle differenti combinazioni, anche con dei livelli di Silhoutte leggermente inferiori, può portare ad avere dei risultati differenti.

Bibliografia

[1] Asadi, Amir Reza & Norouzi, Hossein. (2019) *Content Marketing and Online Trust: The Case of Promoting Knowledge for Video Game Shoppers.*

[2] C.-M. Chiu (2004) *Towards a hypermedia-enabled and web-based data analysis framework, Journal of Information Science* 30(1) 60–72.

[3] TripAdvisor (2015) *5 Tips Inspired by Our New Traveler. Survey* `https://www.tripadvisor.com/TripAdvisorInsights/w661`

[4] Schuckert, M., Liu, X., Law, R. (2015) *Hospitality and tourism online reviews: recent trends and future directions* J. Travel & Tour. Mark. 32(5), 608–621.

[5] Wu, Yonghui; Schuster, Mike; Chen, Zhifeng; Le, Quoc V.; Norouzi, Mohammad (2016) *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.*

[6] V. Hatzivassiloglou and K.R. McKeown (1997) *Predicting the semantic orientation of adjectives*, Proceedings of the 35th Annual Meeting of the ACL and the 8th Conference of the European Chapter of the ACL (1997) 174–181.

[7] J. Wiebe (2000) *Learning subjective adjectives from corpora* Proceedings of the 17th National Conference on Artificial Intelligence (2000) 735–740.

[8] E. Riloff, J. Wiebe and T. Wilson (2003) *Learning subjective nouns using extraction pattern bootstrapping* Proceeding of the 7th Conference on Natural Language Learning (2003) 25–32.

[9] Volcani, Yanon, Fogel, David B. (2001) *System and method for determining and controlling the impact of text* June 28, 2001.

[10] Turney, Peter (2002) *Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews* Proceedings of the Association for Computational Linguistics. pp. 417–424.

[11] Pang, Bo; Lee, Lillian; Vaithyanathan, Shivakumar (2002) *Thumbs up? Sentiment Classification using Machine Learning Techniques"* Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 79–86.

[12] Pang, Bo; Lee, Lillian (2005) *Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales*. Proceedings of the Association for Computational Linguistics (ACL). pp. 115–124.

[13] Snyder, Benjamin; Barzilay, Regina (2007) *Multiple Aspect Ranking using the Good Grief Algorithm*. Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference (HLT-NAACL). pp. 300–307.

[14] C. Weismayer, I. Pezenka, and C. Gan. (2008) *Aspect-Based Sentiment Detection: Comparing Human Versus Automated Classifications of TripAdvisor Reviews* ENTER, page 365-380. Springer, (2018)

[15] Tun Thura Thet, Jin-Cheon Na, and Christopher S.G. Khoo. 2010. *Aspect-based sentiment analysis of movie reviews on discussion boards*. J. Inf. Sci. 36, 6 (December 2010), 823–848.

[16] Afzaal, M., Usman, M., & Fong, A. (2019) *Predictive aspect-based sentiment classification of online tourist reviews*. Journal of Information Science, 45(3), 341–363.

[17] Brychcín, Tomáš & Konkol, Michal & Steinberger, Josef. (2014) *UWB: Machine Learning Approach to Aspect-Based Sentiment Analysis*. Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014). 10.3115/v1/S14-2145.

[18] Blei, David & Ng, Andrew & Jordan, Michael. (2001) *Latent Dirichlet Allocation*. The Journal of Machine Learning Research. 3. 601-608.

[19] Mukherjee, Arjun. (2018) *Extracting Aspect Specific Sentiment Expressions Implying Negative Opinions*.

[20] Ghosh, A., Li, G., Veale, T., Rosso, P., Shutova, E., Reyes, A., Barnden, J. (2015) *Sentiment analysis of figurative language in Twitter*. In: 9th International Workshop on Semantic Evaluation (SemEval), Co-located with NAACL, Denver, Colorado, pp. 470–478. Association for Computational Linguistics.

[21] Reyes, A., Rosso, P., Buscaldi, D. (2012) *From humor recognition to irony detection: the figurative language of social media*. Data Knowl. Eng. 74, 1–12

[22] Patra, Braja & Mazumdar, Soumadeep & Das, Dipankar & Rosso, Paolo & Bandyopadhyay, Sivaji. (2018). *A Multilevel Approach to Sentiment Analysis of Figurative Language in Twitter*.

[23] Kluyver et al. (2016) *Jupyter Notebooks – a publishing format for reproducible computational workflows* Positioning and Power in Academic Publishing: Players, Agents and Agendas, page 87-90, F. Loizides and B. Schmidt, (2016)

[24] *Anaconda Software Distribution*. Computer software. Vers. 2019.10. Anaconda, Oct. 2019. Web. `https://anaconda.com`.

[25] *FAQ about linking – Are website terms of use binding contracts?* (2002) `www.chillingeffects.org`

[26] QVC Sues Shopping App for Web Scraping That Triggered Site Outage `https://newmedialaw.proskauer.com/2014/12/05/qvc-sues-shopping-app-for-web-scraping-that-allegedly-triggered-site-outage`

[27] QVC Inc v. Resultly, LLC - Complaint (2015) `https://it.scribd.com/doc/249068700/LinkedIn-v-Resultly-LLC-Complaint?secret_passwordpEVKDbnvhQL52oKfdrmT`

[28] QVC Inc. v. Resultly LLC (2015) `https://casetext.com/case/qvc-inc-v-resultly-llc` WL 1187500 (E.D. Pa. March 13, 2015)

[29] `https://www.imdb.com/robots.txt`

[30] `https://firefox-source-docs.mozilla.org/testing/geckodriver`

[31] https://www.theguardian.com/travel/2014/dec/23/italy-fines-tripadvisor-500000

[32] http://norvig.com/spell-correct.html

[33] Damerau FJ (1964). *A technique for computer detection and correction of spelling errors*. Commun ACM. 1964;7(3):171–6

[34] Levenshtein VI. (1966) *Binary codes capable of correcting deletions, insertions, and reversals*. Sov Phys Dokl. 1966;10(8):707–10.

[35] https://github.com/wolfgarbe/SymSpell

[36] https://nlp.stanford.edu/software/CRF-NER.shtml

[37] https://nlp.stanford.edu/software/pos-tagger-faq.html#h

[38] George A. Miller, Richard Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller (1993) *Introduction to WordNet: An On-line Lexical Database*

[39] Wordnet, A Lexical Database for English https://wordnet.princeton.edu/

[40] Z. Wu and M. Palmer (1994) *Verbs semantics and lexical selection*. In Proceedings of the 32nd annual meeting on Association for Computational Linguistics, pp. 133-138. Association for Computational Linguistics, 1994, June

[41] Martin Ester, Hans-Peter Kriegel, Jiirg Sander, Xiaowei Xu (1996) *A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise*

[42] https://hdbscan.readthedocs.io/en/latest/faq.html#q-hdbscan-is-failing-to-separate-the-clusters-i-think-it-should

[43] Rennie, Jason & Shih, Lawrence & Teevan, Jaime & Karger, David. (2003). *Tackling the Poor Assumptions of Naive Bayes Text Classifiers*.

[44] Zhang, Harry. (2004). *The Optimality of Naive Bayes*.

[45] Duda, R. O., & Hart, P. E. (1973). *Pattern classification and scene analysis*. Wiley and Sons, Inc.

[46] Cortes, C., Vapnik, V.: *Support vector networks*. Mach. Learn. 20(3), 273–297 (1995)

[47] Vapnik, V.: *Statistical Learning Theory*. Wiley, New York (1998)

[48] Weston, J., Watkins, C.: *Multi-class support vector machines*. In: Proceedings of ESANN99 (1999)

[49] Cramer, J. S. (2002). *The origins of logistic regression*.

[50] Tveit, Amund. (2003). *On the Complexity of Matrix Inversion*.

[51] Wolfe P. *Convergence conditions for ascent methods*. SIAM Rev. 1969; 11(2): 226–35.

[52] Nocedal J, Wright SJ. *Numerical optimization*. 2nd ed. New York: Springer; 2006.

[53] Najafabadi, M.M., Khoshgoftaar, T.M., Villanustre, F. et al. *Large-scale distributed L-BFGS*. J Big Data 4, 22 (2017). https://doi.org/10.1186/s40537-017-0084-5

[54] Xing EP, Ho Q, Xie P, Wei D. *Strategies and principles of distributed machine learning on big data*. Engineering. 2016;2(2):179–95.

[55] Schmidt, M., Le Roux, N. & Bach, F.*Minimizing finite sums with the stochastic average gradient*. Math. Program. 162, 83–112 (2017).

[56] Polyak, Boris T.; Juditsky, Anatoli B. (1992). *Acceleration of stochastic approximation by averaging*. SIAM J. Control Optim. 30 (4): 838–855

[57] `https://scikit-learn.org/stable/modules/linear_model.html #logistic-regression`

[58] Bottou, Léon; Bousquet, Olivier (2012). *The Tradeoffs of Large Scale Learning*. In Sra, Suvrit; Nowozin, Sebastian; Wright, Stephen J. (eds.). Optimization for Machine Learning. Cambridge: MIT Press. pp. 351–368

[59] Hutto, C.J. & Gilbert, Eric. *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*. Proceedings of the 8th International Conference on Weblogs and Social Media, ICWSM 2014.

[60] https://medium.com/analytics-vidhya/simplifying-social-media-sentiment-analysis-using-vader-in-python-f9e6ec6fc52f

[61] https://github.com/cjhutto/vaderSentiment

[62] https://github.com/nltk/nltk/blob/develop/nltk/sentiment/vader.py

[63] https://colab.research.google.com/

[64] https://scikit-learn.org/stable/faq.html#will-you-add-gpu-support

[65] http://rikunert.com/SMOTE_explained

[66] R. A. Fisher (1936). *The use of multiple measurements in taxonomic problems.* Annals of Eugenics. 7 (2): 179–188.