Dipartimento di Impresa e Management
Corso di Laurea Magistrale in Marketing
Cattedra di Customer Intelligence e Logiche di Analisi dei Big Data

SENTIMENT ANALYSIS: ALGORITMI DI CLUSTERING E METODI DI CLASSIFICAZIONE SUPERVISIONATA

RELATORE
Chiar.mo Prof. Luigi Laura

CANDIDATO
Giovanni Luca Cascio Rizzo
Matr. 681701

CORRELATORE
Chiar.mo Prof. Giuseppe Francesco Italiano

ANNO ACCADEMICO 2017/2018

Alla mia famiglia

Sommario

L'analisi dei dati testuali rappresenta oggi, per le aziende, uno degli orizzonti più importanti, sia in termini di volume che di rilevanza delle informazioni ottenibili. Con la crescente disponibilità e popolarità di risorse ricche di opinioni come i social media e i siti di recensione, sorgono nuove opportunità e sfide poichè è possibile utilizzare proattivamente le tecnologie dell'informazione per cercare e comprendere le opinioni altrui. In un mercato che offre un ventaglio sempre più ampio di scelte, le aziende, quanto mai preoccupate per la *customer loyalty* e la propria *web reputation*, necessitano di sistemi automatizzati per accedere alle informazioni di proprio interesse in modo semplice, rapido ed intuitivo.
Questo lavoro di ricerca offre metodi e approcci per la categorizzazione dei documenti e il rilevamento del *sentiment* all'interno di un corpo di recensioni disponibile online. Il focus volge su algoritmi di *machine learning* diretti a scoprire *cluster* naturali di *item*, identificare distribuzioni insite nell'insieme dei dati a disposizione ed offire metodi di classificazione automatica dopo aver individuato, quantificato e caratterizzato il contenuto sentimentale dell'unità di testo.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

L'analisi dei dati testuali rappresenta oggi, per le aziende, uno degli orizzonti più importanti, sia in termini di volume che di rilevanza delle informazioni ottenibili. Con la crescente disponibilità e popolarità di risorse ricche di opinioni come i social media e i siti di recensione, sorgono nuove opportunità e sfide poichè è possibile utilizzare proattivamente le tecnologie dell'informazione per cercare e comprendere le opinioni altrui. In un mercato che offre un vefntaglio sempre più ampio di scelte, le aziende, quanto mai preoccupate per la *customer loyalty* e la propria *web reputation*, necessitano di sistemi automatizzati per accedere alle informazioni di proprio interesse in modo semplice, rapido ed intuitivo. Oggigiorno, i dati vengono generati in una varietà di formati e ad una velocità che non mostra segni di rallentamento. Identificare le informazioni che possono facilitare il processo decisionale tra migliaia di documenti, pagine web e *social media feed* rappresenta un processo complesso e dispendioso in termini di tempo. Per tale motivo, l'informazione in forma testuale viene spesso ignorata o utilizzata solo parzialmente per supportare i processi organizzativi. In quest'ottica, le tecniche di *text mining* ed *opinion mining* aiutano a "digerire" i dati di testo in modo più efficiente offrendo l'opportunità di scoprire modelli interessanti e la capacità di trasformare le informazioni testuali in conoscenze utili per il *reputation management* dell'azienda[1]. Questo lavoro di ricerca si propone di rispondere a tale necessità, fornendo metodi e approcci per la categorizzazione automatica dei documenti e il rilevamento del *sentiment* all'interno di un corpo di recensioni disponibile online.

La prima sezione offre una panoramica sulle techine di *text mining* e *data mining*, in grado di identificare fatti, relazioni ed affermazioni che altrimenti rimarrebbero sepolti sotto una massa di dati testuali. In particolare, tramite l'impiego di algoritmi di *Natural Language Processing* si mostra come decifrare le ambiguità del linguaggio naturale e convertire dati testuali non strutturati in metadati.

[1]Expert System (2017), "Text mining and analytics: how does it work?". Accessibile da: *http://www.expertsystem.com/text-mining-analytics-work/*.

La seconda sezione volge uno sguardo verso la disciplina dell'*Artificial Intelligence* cercando di definire cosa consente ad un sistema, quale uno *smart speaker*, di mostrare attività intelligente. In particolare, la trattazione riguarda i due principali metodi di apprendimento: gli algoritmi di *machine learning* che "allenano" i sistemi di intelligenza artificiale e il *deep learning* che permette di emulare la mente dell'uomo tramite reti neurali artificiali progettate *ad hoc*.

La terza sezione, infine, propone metodi di analisi statistica applicati tramite il linguaggio R e volti a decifrare e comprendere le informazioni contenute nel corpo di recensioni Amazon dello *smart speaker* Amazon Echo. Nel dettaglio, viene offerta una guida alla programmazione standard che va dal *web scraping*, attraverso cui convertire HTML *tag* in metadati poi memorizzati ed analizzati in locale all'interno di un database fino all'approccio *Bag of Words*, in cui ogni recensione viene trasformata in un vettore impiegabile come dato di input (o output) per algoritmi di *machine learning*. Due tecniche di *unsupervised machine learning*, il *Topic Modeling* e gli algoritmi di *clustering*, hanno permesso di identificare distribuzioni insite nell'insieme dei dati a disposizione evidenziando, in modi differenti ma al contempo equipollenti, quattro topic discussi all'interno della raccolta: l'utilizzo domestico del prodotto, la capacità del *device* di riprodurre musica, la qualità del prodotto e la semplicità ed il piacere derivante dall'utilizzo del prodotto. L'elaborato di tesi mostra anche come calcolare la *polarity* della raccolta estraendo il *sentiment* dal testo tramite tre lessici che, per quanto forniscano risultati differenti in senso assoluto, mostrano comunque traiettorie relative simili: un *overall sentiment* assolutamente positivo. Infine, dopo aver quantificato e caratterizzato il contenuto sentimentale dell'unità di testo, sono stati costruiti tre *classifier* predittivi di testo in grado di etichettare correttamente una nuova recensione scritta per il prodotto come positiva o negativa. Per ognuno dei tre modelli (regressione logistica, CART e Random Forest) è stata valutata l'accuratezza e la capacità di generalizzazione ad un set di dati indipendenti. Da una valutazione del *trade-off* tra accuratezza e complessità del modello e dopo un confronto con l'abilità predittiva della *baseline*, si deriva che il CART mostra performance più elevate, poichè più degli altri minimizza il costo di errata classificazione. Per testare il rischio di *overfitting* del CART è stata infine applicata una *k*-fold cross validation i cui risultati mostrano che il modello spiega la devianza presente nei dati originari senza adattarvisi in modo eccessivo.

Capitolo 1

Tecniche di Text Mining per il rilevamento automatico delle informazioni

Una parte importante del nostro comportamento di raccolta delle informazioni è sempre stata quella di scoprire cosa pensano gli altri. Con la crescente disponibilità e popolarità di risorse ricche di opinioni come i social media ed i siti di recensione, sorgono nuove opportunità e sfide poichè è possibile utilizzare proattivamente le tecnologie dell'informazione per cercare e comprendere le opinioni altrui. L'improvvisa esplosione di attività nell'area dell'opinione pubblica si è verificata, in buona parte, come diretta risposta all'impulso di interesse nei nuovi sistemi e nelle capacità di questi ultimi di fornire un mezzo per trattare direttamente con le opinioni quali oggetto di prima classe. In questo contesto si inserisce il *text mining* che, come vedremo nella sezione 1.1, applica tecniche di *data mining* in grado di identificare fatti, relazioni ed affermazioni che altrimenti rimarrebbero sepolti sotto una massa di dati testuali non strutturati. Queste informazioni vengono estratte e trasformate in dati strutturati tramite l'impiego di algoritmi di *Natural Language Processing* che, come più approfonditamente discusso nella sezione 1.3, consentono al computer di "leggere" ed analizzare le informazioni testuali, riconoscendo concetti simili, anche se espressi in modi differenti. Dal momento che sul mercato l'informazione rappresenta una risorsa strategica, le aziende sono sempre più interessate alle opinioni del consumatore. Difatti, in un mondo che offre un ventaglio di scelte sempre più ampio, le aziende sono quanto mai preoccupate per la propria *web reputation* e quindi alla ricerca di nuovi modi per aumentare la *customer loyalty*. In quest'ottica,

la *sentiment analysis*, oggetto della sezione 1.2, si occupa del trattamento computazionale dell'opinione, del *sentiment* e della soggettività del testo nei confronti di entità e loro *feature*, fornendo tecniche ed approcci che promettono di attivare sistemi di ricerca di informazione orientati all'opinione pubblica[1].

1.1 Text Mining

Oggigiorno, i dati vengono generati in una varietà di formati e ad una velocità che non mostra segni di rallentamento. Identificare le informazioni che possono facilitare il processo decisionale tra migliaia di documenti, pagine web e *social media feed* rappresenta un processo complesso e dispendioso in termini di tempo. Per tale motivo, l'informazione in forma testuale viene spesso ignorata o utilizzata solo parzialmente per supportare i processi organizzativi. In quest'ottica, le tecniche di *text mining* aiutano a "digerire" i dati di testo in modo più efficiente poichè offrono l'opportunità di scoprire modelli interessanti e la capacità di trasformare le informazioni testuali in conoscenze utili per implicazioni manageriali[2]. Per descrivere il *text mining*, oggetto di discussione in questa sezione, Oxford fornisce una definizione paradigmatica: "il text mining rappresenta il processo o la pratica di esaminare grandi raccolte di risorse scritte per generare nuove informazioni[3]".

Prima di andare avanti con la trattazione, è necessaria una precisazione volta a definire cosa si intende per dati strutturati, dati semi-strutturati e dati non strutturati. I dati strutturati[4] sono tutti quei dati che possono essere memorizzati in un database SQL (*Structured Query Language*). Sono informazioni altamente organizzate che vengono caricate ordinatamente in un database relazionale e possono essere facilmente mappate in campi pre-progettati. I dati non strutturati hanno una proprio scheletro interno, non conforme a un foglio di calcolo o a un database relazionale. La sfida fondamentale è riuscire a predisporre questa fonte di dati per un uso analitico. I dati semi-strutturati, infine, sono una tipologia di dati strutturati che, per quanto non conformi all'impianto formale dei database relazionali o ad

[1] Pang, B. e Lillian, L. (2008), "Opinion mining and sentiment analysis" in *Foundations and Trends® in Information Retrieval* 2.1–2, 1-135.

[2] Expert System (2017), "Text mining and analytics: how does it work?". Accessibile da: *http://www.expertsystem.com/text-mining-analytics-work/*.

[3] Oxford Reference (2018), "Text Mining". Accessibile da: *http://www.oxfordreference.com/view/10.1093/oi/authority.20110803103357444*.

[4] Elmasri, R. e Navathe, S. (2010), "Fundamentals of database systems". Addison-Wesley Publishing Company.

altre tabelle di dati, contengono *tag* o altri indicatori per separare elementi semantici e definire campi e gerarchie di *record*. Esempi di dati semi-strutturati sono XML, JSON, file csv.

Una volta compreso in che formato si può presentare un dato informatico, è possibile delineare un *mapping* sulla struttura odierna delle informazioni e su come le aziende rispondono a tale stato delle cose. La Figura 1.1 [5] mostra come circa l'80% delle informazioni si presenti in forma non strutturata a fronte di un restante 20% costituito cumulativamente da dati strutturati e semi-strutturati. Poichè il *forecast* al 2020 prevede un aumento della mole di dati non strutturati sia in valore relativo (85-90% del totale) che assoluto (40000 exabytes), le aziende cercano di affrontare questo immenso archivio dati tramite il *data mining* e le nuove applicazioni di *text mining*, con l'obiettivo di creare valore aziendale.

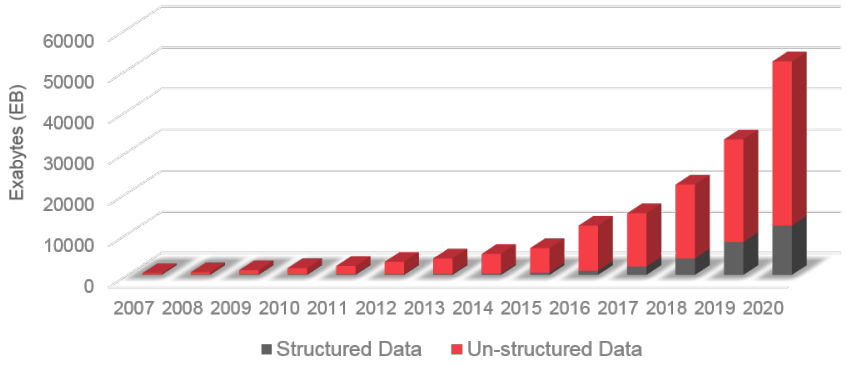

Figura 1.1. Previsione di distribuzione del formato dei dati online al 2020

Sebbene spesso il fine perseguito da entrambe le tecniche sia il medesimo, ovvero sfruttare le informazioni per la scoperta della conoscenza, il *text mining* ed il *data mining* variano in modo significativo nei vincoli di complessità dei dati, tempi di implementazione ed applicazione pratica. A tal proposito, per delineare le differenze, diamo uno sguardo da vicino.

Il *text mining* rappresenta l'insieme dei processi necessari per trasformare risorse testuali non strutturate in informazioni strutturate[6]. Ciò richiede tecniche linguistiche e statistiche in grado sia di analizzare formati di testo libero che di combinare ciascun documento con metadati che possiamo considerare una sorta di ancoraggio

[5]Edureka (2018), "Growth of Unstructured Data – Learn Hadoop". Accessibile da: *https://www.edureka.co/blog/10-reasons-to-learn-hadoop/growth-of-unstructured-data-learn-hadoop-edureka/*.

[6]Feldman, R. e Sanger, J. (2007), "The text mining handbook: advanced approaches in analyzing unstructured data". Cambridge university press.

nella strutturazione. Una volta identificata l'informazione, la stessa può essere automaticamente classificata, indirizzata, riepilogata e visualizzata attraverso un *mapping*. Il *data mining*, invece, consiste nell'applicare algoritmi volti ad estrarre ed analizzare informazioni contenute nel complesso dei dati a disposizione, caricandoli poi in un *data warehouse*[7]. I sistemi di *data mining*, a differenza di quelli di *text mining*, analizzano essenzialmente dati quantitativi omogenei, universali e di facile accesso che non richiedono la comprensione del contesto informativo. In questo modo, una volta definiti gli algoritmi, la soluzione può essere rapidamente implementata. La tecnica è generalmente utilizzata per rilevare automaticamente modelli e relazioni nascoste nei dati o, in alternativa, per finalità predittive. Gli strumenti di *text mining*, invece, devono affrontare sfide tecniche importanti, quali il formato ed il dominio di documenti eterogenei, testi multilingue, abbreviazioni e forme gergali tipiche del linguaggio naturale. La complessità dei dati elaborati rende i progetti di *text mining* più lunghi da implementare. Già solo l'estrazione del testo, infatti, conta diversi stadi linguistici intermedi di analisi quali: individuazione della lingua, *tokenization*, segmentazione, analisi morfo-sintattica, *disambiguation*, riferimenti incrociati ecc. I passaggi successivi all'estrazione dei termini e relativi all'associazione dei metadati concernono una precisa strutturazione del contenuto in modo da poter performare applicazioni specifiche del dominio.

Text mining e *data mining* sono oggi considerate due tecniche complementari. Difatti, nello specifico, il *text mining* estrae, a partire da informazioni testuali, indici numerici significativi tali da rendere le informazioni contenute nel testo accessibili ai vari algoritmi di *data mining*. Queste informazioni possono essere estratte per ricavare un *summary* delle parole contenute nei documenti o per elaborare un *summary* dei documenti in base alle parole in essi occorrenti. A livello più semplice, tutte le parole trovate nei documenti di input vengono indicizzate e contate per costruire una tabella di documenti e parole, cioè una matrice di frequenze che enumera l'occorrenza di ogni parola in ciascun documento. Questo processo base, come approfonditamente discusso nel capitolo 3, può essere ulteriormente perfezionato per escludere alcune parole comuni (*stopword list*) e combinare diverse forme grammaticali della stessa parola (*stemming*). Tuttavia, una volta che dai documenti è stata derivata una matrice dei dati (tabella di parole uniche), è possibile applicare tutte le tecniche statistiche e di *data mining* standard volte a ricavare dimensioni insite nella raccolta testuale o indirizzate ad identificare *keyword* che meglio prevedono

[7] Piatetsky-Shapiro, G. (1996), "Advances in knowledge discovery and data mining" (Vol. 21). U. M. Fayyad, P. Smyth, & R. Uthurusamy (Eds.). Menlo Park: AAAI press.

un'altra variabile di risultato. In sintesi, le fasi del processo di *text mining*, illustrate nella Figura 1.2 tratta da DataFlair[8], sono:

1. Definizione degli obiettivi e acquisizione dei documenti.
2. Codifica dei dati, ovvero costruzione del processo che porta dal documento alla matrice dei dati da analizzare. Tale step si estrinseca in due momenti: la scelta delle unità di analisi, in cui vengono definite le regole che permettono di isolare dal *text corpus* le unità da analizzare (termini, *token*, occorrenze) e il sistema di pesi da adottare, in cui i documenti vengono trasformati in vettori numerici.
3. Estrazione delle informazioni e *word normalization*.
4. Analisi statistiche tramite gli strumenti di *data mining*.
5. Valutazione ed interpretazione dei risultati.
6. Applicazione delle scoperte[9].

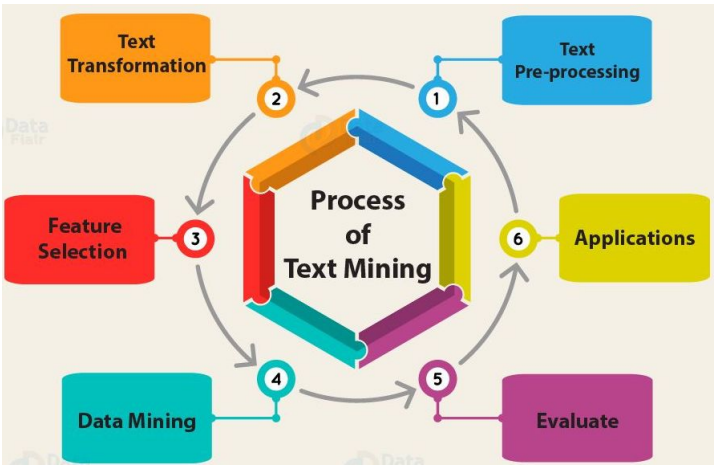

Figura 1.2. Processo di text mining

Tipicamente, il *text mining* viene utilizzato per due funzioni principali: classificare i dati per segmento e prevedere il comportamento. Quattro funzionalità comunemente accettate e meglio approfondite nel capitolo 3 sono[10]:

- *Cleaning*, processo volto ad organizzare il materiale per l'analisi. Tipicamente

[8]DataFlair (2018), "Text Mining in Data Mining-Concepts, Process & Applications". Accessibile da: *https://data-flair.training/blogs/text-mining/*.

[9]Iezzi F. (2018), "Analisi statistica di dati testuali". Accessibile da: *http://didattica.uniroma2.it/assets/uploads/corsi/39157/Analisi_di_dati_testuali.pdf*.

[10]Koslowsky, S. (2010), "Database: Text Mining in Marketing". Accessibile da: *https://www.targetmarketingmag.com/article/text-mining-database-marketing-customer-service-analysis/all/*.

la fase di *cleaning* può richiedere molto tempo, poichè i dati di testo si possono presentare in formati insoliti.

- *Extraction*, in cui estrarre *keyword* comporta la scelta delle parole e delle frasi che meglio identificano la natura del testo. La forma più complessa di estrazione implica un processo che incorpora l'analisi della frase in relazioni soggetto-verbo-oggetto.

- *In categorization*, in cui il testo è classificato meccanicamente in una o più categorie. Se le categorie sono predefinite, è possibile utilizzare algoritmi di *supervised machine learning* per apprendere le sfumature nel testo che distinguono i gruppi predefiniti. Se le categorie sono sconosciute in principio, è possibile utilizzare procedure di *unsupervised machine learning* per classificare il testo in gruppi.

- *Modeling*, in grado di coniugare componenti essenziali del testo con attributi di dati più strutturati tipicamente utilizzati nel *data mining*. Lo scopo di questa unione è ottimizzare ulteriormente la previsione o, semplicemente, fornire una descrizione più dettagliata.

Come è possibile notare, su questa scia il *text mining* è un pò come un cugino di primo grado per il *data mining*: entrambi cercano di discernere modelli e tendenze da rilevanti archivi di dati. Tuttavia non è facile ideare sistemi di computer in grado di "leggere" il testo scritto in forma di linguaggio naturale. Per fortuna, una disciplina nota come *Natural Language Processing*, discussa nella sezione 1.3, sta fornendo storie di successo. Sebbene il *text mining* ed il *data mining* siano oggi considerate tecniche complementari necessarie per una gestione aziendale efficiente, gli strumenti del primo stanno diventando sempre più importanti. Il linguaggio naturale non sarà mai facile da gestire come le cifre, ma il *text mining* è ora più maturo e la sua associazione con il *data mining* ha più senso.

In conclusione, visto il suo largo impiego, il *data mining* è considerato, da decenni, una tecnologia collaudata, robusta ed industriale. Il *text mining*, invece, è stato storicamente pensato come complesso, specifico del dominio, specifico della lingua, sensibile e sperimentale. In altre parole, poichè non è stato compreso abbastanza da avere un supporto gestionale, non è neanche stato mai valutato come un must. Tuttavia, con l'avvento della digitalizzazione, l'aumento dei *social network* e l'incremento della connettività, le aziende sono più preoccupate per la propria *web reputation* e sono alla ricerca di nuovi modi per aumentare la *customer loyalty* in un

mondo che offre un ventaglio di scelte sempre più ampio. Le aziende hanno capito che l'informazione è una risorsa strategica e che il *text mining* non è più un lusso, bensì una necessità. In quest'ottica, l'analisi delle opinioni è divenuto il suo nuovo obiettivo.

1.2 Opinion Mining

Come sappiamo, il web ha cambiato il nostro modo di comunicare. Oggi molte persone si scambiano opinioni, emozioni, idee e critiche sulle piattaforme online, poiché si sentono più libere di esprimere e condividere i propri pensieri ed opinioni riguardo ad un particolare oggetto. La crescente rilevanza dell'analisi del *sentiment* coincide proprio con il fermento creatosi attorno ai social media, siti di recensione, forum di discussione e blog. Il motivo per cui l'*opinion mining* (o *sentiment analysis*) abbia recentemente ricevuto un grande interesse da parte della comunità accademica e delle società commerciali deriva dalla sua capacità di fornire una serie di strumenti adatti per analizzare l'opinione pubblica su diversi argomenti. La disciplina, oggetto di discussione in questa sezione, si focalizza sullo sviluppo di metodi in grado di rilevare automaticamente le informazioni contenute in un testo e determinare la polarità dell'opinione nei confronti di un'entità specifica. Come collocazione disciplinare, possiamo definire l'*opinion mining* come un ibrido tra *information retrieval* e linguistica computazionale in quanto prevede l'applicazione di tecniche che estraggono le informazioni soggettive dal materiale testuale a disposizione concentrandosi non sull'argomento che il documento tratta bensì sull'opinione che il documento stesso esprime[11].

Ma cosa è un opinione e qual è il suo obiettivo? Oggetto di un'opinione è solitamente un'entità denominata, come un'organizzazione, un individuo o un evento. Più formalmente, Liu[12] definisce un'opinione come "una dichiarazione soggettiva, una visione, un'attitudine, un'emozione o una valutazione di un'entità o di un aspetto di un'entità da un detentore di un'opinione". La stessa definizione afferma che "un'entità è un oggetto concreto o astratto come prodotto, persona, evento, organizzazione che può essere rappresentato come una gerarchia di componenti,

[11] Vinodhini, G. e Chandrasekaran, R. M. (2012), "Sentiment analysis and opinion mining: a survey" in *International Journal*, 2(6), 282-292.

[12] Liu, B. (2012), "Sentiment analysis and opinion mining" in *Synthesis lectures on human language technologies*, 5(1), 1-167.

sottocomponenti e i loro attributi[13]".

Definito il concetto di opinione, capiamo ora quali sono i risvolti della *sentiment analysis* nei settori di business. Ciò che le persone scrivono su prodotti, istituti o individui ha un valore importante nella nostra società e risulta utile per qualsiasi azienda o istituzione che si prenda cura del controllo sulla qualità. Difatti, con la crescita esplosiva del web, individui ed organizzazioni utilizzano sempre più le opinioni pubbliche integrandole nel processo decisionale. Oggi disponiamo di un enorme volume di dati censurati registrati in forma digitale, in un mondo in cui le opinioni sono fondamentali per quasi tutte le attività umane e rappresentano fattori chiave per influenzare i nostri comportamenti. Per questo motivo, oggi, le aziende sono quantomai preoccupate per la propria reputazione online e il rilevamento automatico del *sentiment* può rappresentare un ottimo strumento di monitoraggio. L'obiettivo principale della *web sentimet analysis* è, così, cogliere i punti di forza e di debolezza di un'azienda, di un prodotto/servizio o di un brand in generale. Questa attività risulta ad oggi fondamentale se si vuole essere competitivi nel proprio mercato ed evitare crisi di *reputation management*. Alcuni *tool* e piattaforme consentono di monitorare in *real time* tutto quello che accade sul web intorno ad una determinata entità permettendo di comprendere le eventuali motivazioni che stanno minando alla *reputation*; in questo modo è possibile arginare la crisi e limitare i danni. Esistono diversi strumenti che consentono il *tracking* dei sentimenti sui social media e sul web come, per esempio: Meltwater, Google Alert, People Browser, Google Analytics, HootSuite, Tweetstats, Facebook Insights, Pagelever e Marketing Grader di Hubspot. In sintesi, la *sentiment analysis* permette di:

- Analizzare la *web reputation*: possono crearsi situazioni in cui anche un solo cliente insoddisfatto può dar vita ad una maratona di insoddisfazioni in rete verso il brand.

- Comprendere la percezione online di un brand/prodotto/personaggio: i commenti che gli utenti pubblicano in rete si diffondono rapidamente, alimentando dei contenuti che verranno poi condivisi e segneranno in qualche modo il brand in questione (sia positivamente che negativamente).

- Misurare il ritorno delle attività di *social media marketing*: in seguito alle attività sui social media, può risultare complicato tener traccia di tutte le opinioni a riguardo e ricavare una visione d'insieme sul tono delle conversazioni.

[13]Varathan, K. D. et al. (2017), "Comparative opinion mining: a review" in *Journal of the Association for Information Science and Technology*, 68(4), 811-829.

Le metriche relative al *sentiment* non dovrebbero essere utilizzate solo per valutare i sentimenti del proprio brand ma anche per scoprire quali brand stanno ottenendo il massimo coinvolgimento sui social media, quali sono i topic più discussi nei diversi settori e, soprattutto, quali sono i nuovi *need* dei consumatori. Utilizzare nel modo giusto gli strumenti a disposizione per l'analisi dei sentimenti è importante per ottenere il risultato desiderato. Il compito è tecnicamente impegnativo ma praticamente molto utile: da una parte le aziende vogliono conoscere le opinioni dei consumatori sui loro prodotti e servizi; dall'altra parte, i potenziali clienti vogliono conoscere le opinioni degli utenti esistenti prima di utilizzare un servizio o acquistare un prodotto. Tuttavia, trovare e monitorare i siti di opinione e distillare le informazioni in essi contenute rimane un compito formidabile a causa della proliferazione di diversi siti. Ogni sito contiene, in genere, un volume enorme di testo non sempre facilmente processaile e decifrabile; difatti, il lettore umano medio ha difficoltà nell'identificare i siti pertinenti e nel riassumere accuratamente le informazioni e le opinioni contenute. È noto come l'analisi umana delle informazioni testuali sia soggetta a notevoli *bias*. In genere, le persone riscontrano difficoltà a produrre risultati coerenti quando la quantità di informazioni da elaborare è ampia; ciò a causa delle loro limitazioni fisiche e mentali. Circa le limitazioni fisiche, come sappiamo, per quanto consapevoli che gli *user generated data* siano preziosi in quanto forniscono un vista pubblica su diversi temi, al contempo siamo consci del fatto che la mole di dati è immensa ed esaminarli tutti per estrarre informazioni utili è quasi impossibile. A tal fine, i ricercatori hanno iniziato a sviluppare approcci che possono automaticamente estrarre e analizzare le informazioni contenute all'interno di enormi quantità di dati. Per quanto riguarda le limitazioni mentali, ad esempio, le persone dipendono dalle raccomandazioni online o, comunque, prestano maggiore attenzione alle opinioni che si rivelano coerenti con le proprie preferenze. Da qui la necessità di sistemi automatizzati di estrazione e di riepilogo del *sentiment*. Questo perchè i pregiudizi soggettivi ed i limiti mentali possono essere superati solo con un sistema obiettivo di analisi dell'opinione pubblica[14]. Nell'analisi del *sentiment*, esistono alcuni campi di ricerca predominanti:

- La classificazione del *sentiment*, che riguarda la classificazione di interi documenti in base alle opinioni verso determinate entità.

- La classificazione del *sentiment*, basato su *feature*, che considera le opinioni

[14]Liu, B. e Zhang, L. (2012), "A survey of opinion mining and sentiment analysis" in *Mining text data* (pp. 415-463). Springer US.

sulle caratteristiche di determinate entità.

- Il riepilogo delle opinioni, che estrae solo le caratteristiche del prodotto su cui i clienti hanno espresso le proprie opinioni. Il riepilogo delle opinioni è diverso dal riepilogo tradizionale del testo perchè non riassume le recensioni selezionando un sottoinsieme nè riscrive frasi originali delle recensioni per acquisire i principali punti del documento.

Molte delle ricerche sull'analisi del *sentiment* si focalizzano principalmente sulla polarità delle recensioni scritte degli utenti. In questi studi, l'analisi del *sentiment* è spesso condotta su uno dei tre livelli: a livello di documento, a livello di frase o la livello di attributo. La letteratura indica due tipi di tecniche: algoritmi di *machine learning* e orientamento semantico. Come vedremo nella sezione 1.3 e nel capitolo 3, in questo lavoro di tesi ci concentremo sugli algoritmi di *machine learning* che rilevano il *sentiment* utilizzando le tecniche di elaborazione del *Natural Language Processing*.

1.3 Natural Language Processing

Come visto nella sezione 1.2, la quantità di dati testuali da maneggiare è immensa e, dato il volume, è necessario un certo grado di forza bruta computazionale. Con questo in mente, l'intelligenza artificiale può essere utilizzata con buoni risultati per svolgere gran parte del lavoro di decifrazione. In particolare, l'intelligenza artificiale, più approfonditamente discussa nel capitolo 2, viene utilizzata per scomporre il contenuto testuale in parti componenti (nomi, verbi, parole di *sentiment* ecc.) in modo da poter sviluppare una comprensione del *sentiment* dell'autore. Per molte attività di analisi del testo, ma in particolare per quelle su larga scala dove il testo richiede un'analisi relativamente standardizzata, il *machine learning* può rivelarsi un approccio eccellente perchè in grado di costruire modelli analitici in modo autonomo. In questa sezione entreremo nel merito della questione e cercheremo di capire cosa si intende per algoritmi di *machine learning* e per *Natural Language Processing*, argomenti tanto blasonati quanto spesso descritti superficialmente.

In generale, per *machine learning* si intende l'abilità delle macchine (intese come computer) di apprendere senza essere state esplicitamente e preventivamente programmate[15]. Da una prospettiva informatica, anziché scrivere il codice di pro-

[15]SAS (2018), "L'evoluzione del machine learning". Accessibile da: *https://www.sas.com/it_it/insights/analytics/machine-learning.html*.

grammazione attraverso il quale, passo dopo passo, si "dice" alla macchina cosa fare, al computer vengono forniti set di dati inseriti in un generico algoritmo che sviluppa una propria logica per svolgere la funzione richiesta. L'algoritmo migliora le proprie prestazioni in modo adattivo, ovvero man mano che gli esempi da cui apprendere aumentano producendo, alla fine, risultati e decisioni affidabili e replicabili. In questo senso, l'aspetto più importante è la ripetitività: più i modelli sono esposti ai dati, più sono in grado di adattarvisi in modo autonomo. I principali approcci di *machine learning*, di seguito elencati, sono tre[16]:

- *Supervised machine learning*: include metodi che, dato un insieme di dati di *training*, apprendono una determinata funzione. L'obiettivo di questi algoritmi, noti come *classifier*, è di prevedere l'attributo di classe per i dati non etichettati (dati di *testing*). I classificatori, in pratica, si basano sull'apprendere una serie di funzionalità estratte dai dati di *training*. Tale apprendimento esperienziale viene poi utilizzato dal classificatore per formulare inferenze future.

- *Unsupervised machine learning*: include algoritmi di apprendimento utilizzati per rilevare la struttura nascosta nei dati non etichettati. In particolare, i metodi consistono nel fornire al sistema informatico una serie di input che il sistema stesso riclassificherà ed organizzerà sulla base di caratteristiche comuni con il fine di predire su input successivi.

- *Reinforcement learning*: include algoritmi in grado di apprendere ed adattarsi alle mutazioni dell'ambiente. Questa tecnica di programmazione si basa sul presupposto di poter ricevere degli stimoli dall'esterno a seconda delle scelte dell'algoritmo. Una scelta corretta comporterà un premio mentre una scelta inesatta porterà ad una penalizzazione del sistema. L'obiettivo è raggiungere il maggior premio possibile e, di conseguenza, il migliore risultato possibile.

L'area del *machine learning* dedicata al senso della parola scritta è nota come *Natural Language Processing* (da ora in poi NLP). Come visto nella sezione 1.1, la *sentiment analysis* è una delle aree di ricerca più attive del NLP ed è ampiamente studiata ed applicata nell'ambito del *data mining*, del *web mining* e del *text mining*. Questo perchè gli algoritmi di NPL consentono al computer di "leggere" ed analizzare le informazioni testuali, riconoscendo concetti simili, anche se espressi in modi differenti. Il NPL, in pratica, interpreta il significato del testo ed identifica, analizza e sintetizza fatti e relazioni rilevanti che rispondono direttamente alla domanda del

[16]Lison, P. (2015). "An introduction to machine learning".

consumatore. Il successo del NLP è dovuto al fatto che, negli ultimi anni, con il fenomeno Internet ed il fermento dei nuovi social media, l'uso della lingua scritta sta crescendo esponenzialmente. La diretta conseguenza è una mole immensa di *user generated data* codificati in linguaggio naturale, che solo i software NLP più avanzati sono in grado di gestire[17]. A tal proposito, è necessaria una precisazione: per linguaggio naturale si intende quello umano, principalmente per poterlo distinguere da altri linguaggi, detti artificiali. A differenza dei linguaggi artificiali come quelli di programmazione e le notazioni matematiche, i linguaggi naturali si sono evoluti passando da una generazione all'altra e sono difficili da definire con regole esplicite. In tale contesto si inseriscono, appunto, i metodi di NPL, tramite cui il testo può essere estratto in modo sistematico, completo, riproducibile consentendo a che le informazioni aziendali critiche vengano acquisite e raccolte automaticamente[18]. Tali metodi possono essere applicati per analizzare la lingua a due diversi livelli: a livello sintattico e a livello semantico. L'analisi sintattica esamina la sintassi delle frasi; l'analisi semantica, invece, ha lo scopo di identificare ed analizzare il significato di parole e frasi. Da un punto di vista computazionale, però, la manipolazione del linguaggio naturale da parte del computer potrebbe rilevarsi particolarmente ostica. Se ad un estremo potrebbe essere semplice come contare le frequenze di occorrenza di una parola per confrontare diversi stili di scrittura, all'altro estremo il NLP comporta la comprensione di espressioni umane complete. Per marginare tali problemi, Il NPL impiega metodologie volte a decifrare le ambiguità del linguaggio umano: riepilogo automatico, *tag* di parte del discorso, *disambiguation*, *entity extraction*, *relations extraction*, nonché comprensione e riconoscimento del linguaggio naturale. Per funzionare, qualsiasi software di NPL ha bisogno di una base di conoscenza coerente, ovvero un lessico di parole, un set di dati per le regole linguistiche e grammaticali, un'ontologia ed entità aggiornate. Possiamo quindi dire che "fornendo più interfacce uomo-macchina e un accesso più sofisticato alle informazioni archiviate, l'elaborazione del linguaggio ha assunto un ruolo centrale nella società dell'informazione multilingue[19]".

In quest'ottica, le tecnologie basate sul NLP stanno diventando sempre più diffuse e il loro impiego è trasversale a più settori. Oggi, infatti, il software NPL è

[17]Celi (2018), "Natural Language Processing, Software e risorse per l'elaborazione linguistica". Accessibile da: *https://www.celi.it/tecnologia/natural-language-processing/*.

[18]Linguamatics (2018), "NLP text mining–the ten thousand foot view". Accessibile da: *https://www.linguamatics.com/what-is-text-mining-nlp-machine-learning*.

[19]Bird, S. et al. (2009), "Natural language processing with Python: analyzing text with the natural language toolkit", *O'Reilly Media, Inc.*

un processo ombra che viene eseguito in *background* su molte applicazioni comuni. Ad esempio, telefoni e computer supportano il riconoscimento del testo e della scrittura a mano, i motori di ricerca web danno accesso a informazioni rinchiuse in testo non strutturato, la traduzione automatica ci permette di recuperare testi scritti in una lingua e leggerli in un'altra. Un ulteriore impiego di tali tecniche riguarda i cosiddetti assistenti virtuali, oggetto di discussione nella sezione 2.3. A consentire tale progresso tecnologico è stato, di certo, l'aumento della capacità predittiva dei modelli statistici del linguaggio naturale registrato negli ultimi decenni. Questi sforzi, pur variando in specifiche, affrontano due compiti essenziali di *modeling* statistico: determinare un insieme di statistiche che catturino il comportamento di un processo casuale (selezione delle funzionalità) e, dato tale insieme di statistiche, allinearle in un modello accurato del processo, in grado di prevedere l'output futuro (selezione del modello). L'applicazione di tali metodologie sarà ampiamente discussa nel capitolo 3.

1.4 Vantaggi, Limiti e Campi di Applicazione

L'analisi del *sentiment* si prefigura come una tecnica efficace nella rilevazione delle opinioni e, poichè le risorse computazionali a disposizione sono in grado di operare su una grande mole di dati, l'*opinion mining* offre enormi opportunità in diversi campi di applicazione. Il suo utilizzo, difatti, fornisce facilitazioni nelle analisi competitive, nelle analisi di marketing e, soprattutto, nella rilevazione di voci sfavorevoli per la gestione del rischio.

Ad oggi, il mondo della ricerca e le aziende prediligono rilevare le preferenze e la *satisfaction* del consumatore all'interno di documenti online invece di analizzare i risultati ottenuti da sondaggi la cui efficacia, di solito, risulta molto limitata. Le cause dell'inefficacia dei sondaggi corrispondono con: difficoltà riscontrate nel creare questionari efficaci, limitazioni dovute alla dimensione del campione di rispondenti ottenuto, "negligenza" con cui i consumatori esprimono le proprie preferenze. Creare un sondaggio per ogni prodotto/funzione progettandone formato, distribuzione, tempi (inviare un modulo subito dopo l'acquisto potrebbe non essere molto informativo), oltre alla dipendenza dalla buona volontà delle persone nel condurre il sondaggio stesso, è un'attività dispendiosa in termini di tempo e risorse economiche, con risultati non sempre accurati. In quest'ottica, l'analisi automatica delle opinioni porta con sè importanti vantaggi. In primo luogo, le persone che condividono le proprio opinioni di solito lo fanno proattivamente offrendo giudizi

più pronunciati rispetto alla media. Tali opinioni influenzano la lettura degli altri utenti, portando al cosiddetto marketing del *word of mouth*. In secondo luogo, le opinioni vengono estratte in tempo reale, consentendo tempi di risposta più rapidi ai cambiamenti del mercato e statistiche dettagliate che consentono di tracciare le tendenze nel tempo[20]. La rilevanza del *text mining*, come ampiamente trattato nella sezione 1.1, deriva dal fatto che il testo non strutturato è molto comune e può rappresentare la maggior parte delle informazioni disponibili per un particolare progetto di ricerca o di *data mining*. Esempi sono:

- Analisi delle risposte al sondaggio aperto: nella ricerca per mezzo di sondaggi, non è insolito includere domande aperte relative all'argomento in esame. L'idea è consentire agli intervistati di esprimere le proprie opinioni senza vincolarle a dimensioni particolari o ad un determinato format di risposta. Ciò potrebbe fornire informazioni sul *sentiment* del consumatore che altrimenti non potrebbero essere scoperto con questionari strutturati.

- Elaborazione automatica di testo: un'altra applicazione comune per il *text mining* è quella di aiutare nella classificazione automatica dei testi. Ad esempio, è possibile filtrare automaticamente la posta indesiderata in base a determinati termini o parole, non propri di messaggi legittimi.

- Analizzare richieste di garanzia o assicurazione, interviste diagnostiche, ecc: in alcuni settori, la maggior parte delle informazioni viene raccolta in forma di *free text flow*, come, ad esempio, le richieste di risarcimento o le interviste mediche. Sempre più spesso questo testo viene raccolto elettronicamente, quindi prontamente disponibile per l'applicazione di algoritmi di *text mining*. Tali informazioni possono essere sfruttate per identificare gruppi comuni di problemi e reclami o, in campo medico, per fornire indizi utili per l'effettiva diagnosi.

- Investigare sui concorrenti tramite *crawling* dei loro siti web: un altro tipo di applicazione potenzialmente utile consiste nell'elaborare automaticamente i contenuti delle pagine web in un determinato dominio. Ad esempio, si può accedere ad una pagina web e iniziare a scansionare i collegamenti che si trovano lì per elaborare tutte le pagine web cui si fa riferimento. In questo modo, è possibile ottenere automaticamente un elenco di termini e documenti disponibili in quel sito e quindi determinare rapidamente i termini e le

[20] Boiy, E. e Moens, M. F. (2009), "A machine learning approach to sentiment analysis in multilingual Web texts" in *Information retrieval*, 12(5), 526-558.

funzionalità più importanti. È facile vedere come queste capacità possano fornire in modo efficiente preziose informazioni di business sulle attività dei concorrenti.

Come visto, la ricerca si è diffusa al di fuori dell'informatica abbracciando innumerevoli settori proprio per l'importanza che l'analisi delle opinioni ha per le imprese e la società nel suo insieme. Nel marketing, in particolare, il rilevamento automatico del *sentiment* aiuta ad individuare, in via preventiva o successiva, discriminanti del successo di una campagna pubblicitaria o del lancio di un nuovo prodotto, determinare quali versioni di un prodotto/servizio sono più popolari e persino ad identificare a quali unità demografiche piacciono o meno specifiche *feature*.

Per quanto possa rivelarsi utile, la *sentiment analysis* presenta anche diverse sfide. La prima è di carattere computazionale: una *opinion word* potrebbe essere considerata positiva in un determinato contesto e negativa in un'altro; ad oggi nessun *tool* che registra ed attribuisce la *polarity* è in grado di cogliere concetti emotivi complessi come l'ironia o il sarcasmo. Una seconda sfida deriva dal fatto che le persone non sempre esprimono le opinioni nello stesso modo e, a volte, possono rivelarsi contraddittorie nelle loro dichiarazioni.

In generale, la *sentiment analysis* è un compito che richiede una profonda comprensione del contesto testuale, attingendo al senso comune e alla comprensione del dominio, nonchè alla conoscenza linguistica. L'interpretazione delle opinioni può essere discutibile anche per gli umani, figuriamoci per un mezzo computazionale. Ad esempio, se volessimo determinare all'interno di una raccolta testuale se ciascun specifico documento è in equilibrio favorevole o sfavorevole nei confronti del prodotto oggetto di discussione, sicuramente riscontreremmo difficoltà nel raggiungere un consenso. Pertanto ci si concentra sulla ricerca di dichiarazioni locali di preferenza piuttosto che sull'analisi della preferenza generale. L'esistenza di affermazioni che esprimono sentimenti è più affidabile rispetto all'opinione generale; in questo modo, invece di analizzare le preferenze dell'intero contesto, cerchiamo di estrarre ogni dichiarazione sulle preferenze e presentarle in modo che l'azienda possa utilizzare i risultati in base a specifiche esigenze applicative. Le nostre convinzioni e percezioni della realtà e le scelte che facciamo sono in gran parte condizionate da come gli altri vedono e valutano il mondo; per questo motivo, quando abbiamo bisogno di prendere una decisione, cerchiamo spesso le opinioni degli altri. Questo è vero non solo per gli individui ma anche per le organizzazioni[21].

[21] Liu, B. (2012), "Sentiment analysis and opinion mining" in *Synthesis lectures on human language technologies*, 5(1), 1-167.

Capitolo 2

Strumenti di Artificial Intelligence: i Virtual Assistant

Definire cosa sia esattamente l'l'*Artificial Intelligence* è un compito arduo. Poichè l'Artificial Intelligence (da ora in poi AI) è un settore relativamente recente e in fortissima evoluzione, non esiste una definizione universalmente accettata. Bellman[1], per esempio, la definisce "l'automazione di attività che associamo al pensiero umano" come "il prendere decisioni, la risoluzione automatica di problemi, l'apprendimento"; per Knight[2] si tratta dello "studio delle facoltà mentali mediante l'uso di modelli computazionali" mentre Luger[3] la identifica con la branca dell'informatica che riguarda l'automazione di comportamenti intelligenti".

Nella sua accezione puramente informatica, l'AI rappresenta la disciplina che racchiude le teorie e le tecniche pratiche (programmazione e progettazione di sistemi hardware e software) per lo sviluppo di algoritmi che consentano alle macchine di mostrare attività intelligente, ovvero dimostrino di possedere caratteristiche considerate tipicamente umane quali, ad esempio, le percezioni visive, spaziotemporali e decisionali. In questo senso, non si tratta solo di intelligenza intesa come capacità di calcolo o conoscenza di dati astratti, ma anche e soprattutto di tutte quelle differenti forme di intelligenza riconosciute dalla teoria di Gardner[4] e

[1]Bellman, R. (1978), "An introduction to artificial intelligence: can computer think?", (No. 04; Q335, B4).

[2]Knight, K. e Marcu, D. (2002), "Summarization beyond sentence extraction: A probabilistic approach to sentence compression" in *Artificial Intelligence*, 139(1), 91-107.

[3]Luger, G. F. e Stubblefield, W. A. (1993), "Artificial intelligence: its roots and scope" in *Artificial intelligence: structures and strategies for*, 1-34.

[4]Gardner, H. E. (2000), "Intelligence reframed: Multiple intelligences for the 21st century". Hachette UK.

che vanno dall'intelligenza spaziale a quella sociale, da quella cinestetica a quella introspettiva. Un sistema intelligente viene realizzato cercando di ricreare una o più di queste differenti forme di intelligenza che, per quanto comunemente considerate ad esclusivo pannaggio dell'uomo, in realtà possono essere ricondotte a particolari comportamenti riproducibili da macchine.
Il problema complesso dello sviluppare sistemi che esibiscano comportamenti intelligenti è stato affrontato operando una scomposizione in sotto-problemi, ognuno con uno specifico ambito di ricerca:

- Deduzione, ragionamento e *problem solving*: l'obiettivo è sviluppare algoritmi che, data una rappresentazione simbolica dello stato delle cose, cercano di raggiungere l'obiettivo desiderato considerando aspetti compessi quali l'incertezza e l'incompletezza delle informazioni.

- Rappresentazione della conoscenza: l'obiettivo è definire quale tipo di conoscenza è necessario (o opportuno) integrare all'interno di un sistema intelligente, e come rappresentare i diversi tipi di informazione (v. sez. 2.3).

- Pianificazione: per permettere ai sistemi intelligenti di prevedere e rappresentare stati futuri del mondo e per prendere decisioni al fine di raggiungere tali stati massimizzando il valore atteso delle azioni, essi devono essere in grado di definire degli obiettivi e di perseguirli.

- *Machine learning*: l'obiettivo è sviluppare algoritmi in grado di migliorare automaticamente le proprie performance attraverso l'esperienza (v. sez. 2.2).

- *Natural Language Processing*: la capacità di elaborare il linguaggio naturale fornisce ai sistemi intelligenti la possibilità di leggere e capire il linguaggio utilizzato dagli esseri umani (v. sez. 1.3).

- Movimento e Manipolazione: l'obieittivo è sviluppare sistemi di robotica.

Prima di addentrarci nel pieno della complessità e delle applicazioni dei sistemi di AI, è necessaria una precisazione sulla realtà odierna. Il presente parla di AI debole, in quanto un computer non è in grado di raggiungere le capacità intellettive umane, ma solo di simularne alcuni processi cognitivi senza però riuscire a riprodurli nella loro totale complessità. I fautori della teoria dell'AI forte, invece, si spingono oltre e ipotizzano che un giorno le macchine saranno dotate di un'intelligenza propria, autonoma e indipendente, pari o superiore a quella umana.

2.1 Cenni Storici sull'AI

Il fermento attuale attorno alla disciplina dell'AI si spiega con la maturità tecnologica raggiunta sia nel calcolo computazionale (oggi ci sono sistemi hardware molto potenti, di ridotte dimensioni e con bassi consumi energetici), sia nella capacità di analisi in *real time* ed in tempi brevi di enormi quantità di dati disponibili in qualsiasi forma[5].

L'interesse della comunità scientifica per l'AI, però, ha inizio da molto lontano. Il primo vero progetto di AI risale al 1943 quando due ricercatori proposero al mondo scientifico il primo neurone artificiale. A questa ricerca seguì la ricerca dello psicologo canadese Donald Olding Hebb, grazie al quale vennero analizzati in dettaglio i collegamenti tra i neuroni artificiali ed i modelli complessi del cervello umano[6]. I primi prototipi funzionanti di reti neurali (algoritmi matematici sviluppati per riprodurre il funzionamento dei neuroni biologici volti alla risoluzione di problemi di AI) arrivarono verso la fine degli anni '50 e l'interesse del pubblico si fece maggiore grazie al giovane Alan Turing che già in quegli anni cercava di spiegare come un computer potesse comportarsi come un essere umano. Per come viene definita oggi, l'AI nasce nel 1956, quando John McCarthy conia il termine "*Artificial Intelligence*" in occasione di un seminario negli USA al quale invitò ricercatori sulla teoria degli automi, sulle reti neurali, sullo studio dell'intelligenza, ma con interessi che spaziavano anche dallo sviluppo di sistemi di ragionamento automatico ai giochi come la dama. Durante questo storico convegno furono presentati alcuni programmi capaci di effettuare ragionamenti logici, in particolar modo legati alla dimostrazione di teoremi matematici. Gli anni successivi furono caratterizzati da grande fermento intellettuale e sperimentale tanto che università e società informatiche (tra le quali l'IBM) puntarono alla ricerca e allo sviluppo di nuovi programmi e software in grado di pensare e agire come gli esseri umani. Su questa scia fu creato Lisp, il primo linguaggio di programmazione che per oltre trent'anni è stato alla base dei software di AI. Tuttavia, se da un lato si riuscirono a sviluppare software sempre più sofisticati in grado di risolvere elaborazioni matematiche, dall'altro lato si iniziarono a notare le prime limitazioni dell'AI, che non sembrava poter riprodurre le capacità intuitive e di ragionamento tipiche degli esseri umani. Durante gli anni '70 divenne sempre più evidente che quanto realizzato fino ad allora non era più sufficiente alle

[5]Boldrini N. (2018), "Cos'è l'Intelligenza Artificiale, perchè tutti ne parlano e quali sono gli ambiti applicativi". Accessibile da *www.ai4business.it/intelligenza-artificiale/intelligenza-artificiale-cose/*.

[6]Hebb, D. O. (1955), "Drives and the CNS (conceptual nervous system)" in *Psychological review*, 62(4), 243.

nuove necessità. L'obiettivo divenne realizzare macchine e programmi in grado di andare oltre la semplice soluzione di teoremi matematici e ricercare soluzioni a problematiche più vicine alla realtà dell'uomo. La sfida, quindi, divenne cercare di riprodurre software che potessero ragionare e prendere decisioni in seguito alla valutazione di differenti possibilità; ovvero soluzioni che potessero variare a seconda dell'evoluzione dei parametri in corso d'opera. Prima di poter essere risolto, questo tipo di problema prevedeva anzitutto la soluzione di uno step propedeutico, ovvero quello di realizzare dei percorsi semantici per le macchine. Nella pratica ciò si sarebbe dovuto tradurre in un linguaggio che permettesse di programmare le diverse possibilità previste da un ragionamento, semplice o complesso che fosse. Il passaggio a questo step si dimostrò particolarmente complesso tanto che la ricerca subì un brusco rallentamento. La nuova era dell'AI si aprì tra gli anni '80 e '90 con il nuovo utilizzo dell'algoritmo che permetteva l'apprendimento per reti neurali; un algoritmo che, già ideato alla fine degli anni '60, non aveva trovato applicazione a causa delle carenze dovute ai sistemi di apprendimento dei primi programmi di AI. La prima svolta importante dal punto di vista tecnologico arrivò con lo sviluppo e l'ingresso sul mercato "allargato" (arrivando cioè al grande pubblico) dei processori grafici Gpu (*Graphics processing unit*), chip di elaborazione dati molto più veloci delle Cpu, in grado di supportare processi complessi molto più rapidamente, per altro operando a frequenze più basse e consumando meno energia rispetto alle Cpu. Il primo emblematico successo dell'AI risale al 1997 e fu quello che vide il confronto tra Deep Blue, una macchina realizzata dalla IBM, ed il campione di scacchi allora in carica Garry Kasparov. I continui miglioramenti apportati al sistema di apprendimento di Deep Blue permisero la vittoria della macchina che dimostrò di aver raggiunto un livello di creatività così elevato da andare oltre le conoscenze del giocatore stesso. In effetti, lo sviluppo delle Gpu ridusse notevolmente i tempi di addestramento delle reti, abbassandoli di 10/20 volte. Con l'ingresso delle Gpu negli anni si sono raggiunti enormi benefici in termini di efficienza e potenza di calcolo; basti pensare al fatto che una Cpu tradizionale è costituita da diversi *core* ottimizzati per l'elaborazione seriale sequenziale mentre una Gpu è dotata di un'architettura parallela con migliaia di *core* progettati per la gestione simultanea di più operazioni[7] (oltretutto di minori dimensioni e più efficienti). L'ondata più recente è arrivata nell'ultimo decennio con lo sviluppo dei cosiddetti "chip neuromorfici", ovvero microchip che integrano elaborazione dati e *storage* in un unico micro-componente

[7] Lu, H. et al. (2018), "Brain intelligence: go beyond artificial intelligence" in *Mobile Networks and Applications*, 23(2), 368-375.

(grazie all'accelerazione che ha avuto la ricerca nel campo delle nanotecnologie) per emulare le funzioni sensoriali e cognitive del cervello umano.

Già da questo rapidissimo viaggio storico si intuisce che dare una definizione esatta di AI è un compito arduo ma, analizzandone le evoluzioni, siamo in grado di tracciarne i contorni e quindi di fare alcune importanti classificazioni.

2.2 Machine Learning, Deep Learning e Reti Neurali Artificiali

Lo sviluppo di algoritmi in grado di migliorare il comportamento della macchina (inteso come capacità di agire e prendere decisioni) apprendendo tramite l'esperienza ha rappresentato, di certo, uno dei principali passi avanti nella storia dell'AI. Costruire algoritmi in grado di imparare dai propri errori è difatti fondamentale per realizzare sistemi intelligenti che operino in contesti per i quali i programmatori non possono prevedere a priori tutte le possibilità di sviluppo. Ciò che caratterizza l'AI, da un punto di vista tecnologico e metodologico, è proprio il metodo di apprendimento con cui l'intelligenza diventa abile in un compito o azione che sia. In particolare, i due modelli di apprendimento oggetto di trattazione in questa sezione sono:

- *Machine learning*, branca dell'AI che riguarda lo studio, la costruzione e l'implementazione di algoritmi matematici attraverso i quali si permette ad una macchina di apprendere autonomamente e, a partire da un insieme di dati in ingresso, a costruire modelli predittivi riducendo il peso degli errori al termine di ogni processo d'apprendimento. In questo modo la macchina potrà svolgere una specifica attività senza che venga preventivamente programmata[8]. In particolare, come già visto nella sezione 1.3, distinguiamo tre approcci: il *supervised machine learning*, dove le etichette sono create dall'addestratore per rendere la macchina capace di scoprire relazioni tra input ed etichette; il *unsupervised machine learning* dove, invece, le etichette non sono disponibili e si chiede alla macchina semplicemente di trovare dei cluster all'interno dei dati e, infine, il *reinforcement learning* che realizza algoritmi in grado di apprendere ed adattarsi alle mutazioni dell'ambiente.

[8]Goldberg, D. E. e Holland, J. H. (1988), "Genetic algorithms and machine learning" in *Machine learning*, 3(2), 95-99.

- *Deep learning*, modello di apprendimento ispirato alla struttura ed al funzionamento del cervello biologico. Se il *machine learning* allena l'AI, il *deep learning* rappresenta l'algoritmo che permette di emulare la mente dell'uomo. In questo caso il *deep learning* necessita, però, sia di algoritmi che di reti neurali artificiali progettate ad hoc (*Deep Artificial Neural Networks*) e di una capacità computazionale in grado di reggere differenti strati di calcolo ed analisi[9].

In sintesi, Il *deep learning* è un caso particolare di *machine learning* e le reti neurali sono i sistemi di calcolo. In quest'ottica, a fronte del viaggio storico intrapreso nella sezione 2.1, è possibile offrire una classificazione che definisce e ripercorre le tappe dell'AI, del *machine learning* e del *deep learning*. Come illustrato nella Figura 2.1, l'AI, in quanto *engineering* di sviluppare programmi e macchine intelligenti contiene la branca del *machine learning*, ovvero l'abilità delle macchine ad apprendere senza che le stesse siano state preventivamente programmate. In questo insieme si inserisce il *deep learning*, dove l'apprendimento avviene tramite reti neurali artificiali[10].

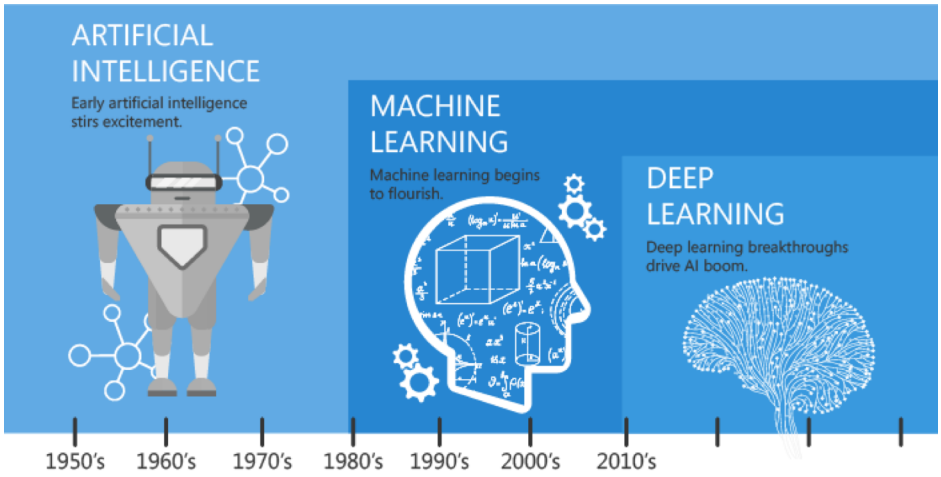

Figura 2.1. Campi dell'Artificial Intelligence

Dopo aver definito i concetti di AI, *machine learning* e *deep learning*, proviamo ad approfondire il discorso sulle reti neurali artificiali, sistemi oggi in uso nel riconoscimento di *pattern*, nel riconoscimento vocale e delle immagini e, per noi più di interese, nei sistemi di *Natural Language Processing* (di cui abbiamo già lungamente discusso nella sezione 1.3). Questo è necessario anche per meglio comprendere il

[9]Schmidhuber, J. (2015), "Deep learning in neural networks: An overview" in *Neural networks*, 61, 85-117.

[10]Neeraj Kumar (2017), "What is the difference between Machine Learning and Deep Learning". Accessibile da: *https://medium.com/Say2neeraj/what-is-the-difference-between-machine-learning-and-deep-learning-5795e4415be9*.

funzionamento del prodotto oggetto di analisi nel capitolo 3, ovvero l'assistente virtuale Alexa incorporato nello *smart speaker* Amazon Echo.

Le reti neurali artificiali sono modelli di calcolo matematico-informatici che cercano di simulare le reti neurali biologiche, ovvero sistemi costituiti da migliaia di interconnessioni tra neuroni (sinapsi) che ci consentono di ragionare e di gestire ogni funzione del corpo[11]. Capiamo come funzione una rete neurale biologica: dopo aver ricevuto dati e segnali esterni, gli stessi vengono elaborati in informazioni attraverso i neuroni (che rappresentano la capacità di calcolo) interconnessi tra loro in una struttura non-lineare e in grado di variare in risposta a tali dati/stimoli esterni. Allo stesso modo, le reti neurali artificiali sono strutture non lineari di dati statistici organizzate come strumenti di modellazione: ricevono segnali esterni su uno strato di nodi d'ingresso (che rappresenta l'unità di elaborazione, il processore); ognuno di questi nodi d'ingresso è collegato a svariati nodi interni della rete che, tipicamente, sono organizzati a più livelli in modo che ogni singolo nodo possa elaborare i segnali ricevuti trasmettendo ai livelli successivi il risultato delle sue elaborazioni (quindi delle informazioni più evolute, dettagliate). In particolare, i nodi vengono dislocati su livelli che, come mostrato nella Figura 2.2, possono essere di tre tipi[12]:

1. Livello di ingresso (*Input Layer*): livello progettato per ricevere le informazioni provenienti dall'esterno al fine di imparare a riconoscere e processare tali informazioni.
2. Livello nascosto (*Hidden Layer*): collega il livello di ingresso con quello di uscita ed aiuta la rete neurale ad apprendere le relazioni complesse analizzate dai dati. Spesso i livelli nascosti sono più di uno.
3. Livello di uscita (*Output Layer*): livello finale che mostra il risultato di quanto il programma è riuscito ad imparare.

Le reti neurali artificiali si presentano, così, come un sistema adattivo, ovvero in grado di modificare la propria struttura adattandola alle specifiche necessità derivanti dalle varie informazioni (dati esterni e informazioni interne) ottenute nelle diverse fasi di apprendimento. Ad ogni connessione tra neuroni è associato un peso che determina l'importanza del valore di input. I pesi iniziali sono impostati casualmente. Ogni neurone, invece, ha una funzione di attivazione che permette di definire un output data una serie di dati di input analizzati dal neurone stesso e,

[11] Schalkoff, R. J. (1997), "Artificial neural networks" (Vol. 1). New York: McGraw-Hill.

[12] Fauske, K. M. (2006), "Neural network". Accessibile da: *http://www.texample.net/tikz/examples/neural-network/*.

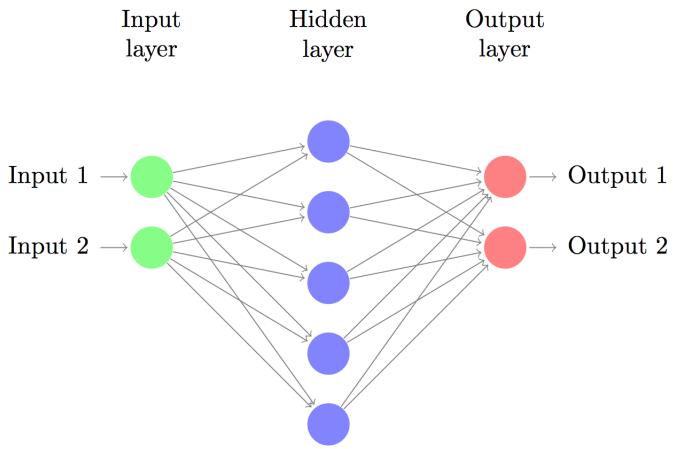

Figura 2.2. Struttura delle reti neurali artificiali

una volta che i dati in ingresso hanno attraversato tutti gli strati della rete neurale, la stessa restituisce i dati di output attraverso il livello di uscita. La parte complessa di una rete neurale è il suo apprendimento. L'apprendimento avviene quando c'è un qualche tipo di feedback, ovvero una risposta che permette di verificare se si è appreso quello che si sta imparando. Più nel dettaglio, le reti neurali sono programmate per imparare allo stesso modo ed utilizzano un algoritmo chiamato *Backpropagation*. Questo algoritmo prevede di confrontare il risultato ottenuto da una rete con l'output che si vuole in realtà ottenere e, usando la differenza tra i due risultati, prevede di modificare i pesi delle connessioni tra i livelli della rete partendo dal livello output[13]. In seguito, procedendo a ritroso, l'algortimo modifica i pesi dei livelli nascosti e infine quelli dei livelli di input. Per far ciò sviluppa una funzione di costo appropriata al problema da risolvere. In definitiva, dal punto di vista matematico, una rete neurale può essere definita come una funzione composta, ovvero dipendente da altre funzioni a loro volta definibili in maniera differente a seconda di ulteriori funzioni dalle quali dipendono. Questo significa che, all'interno di una rete neurale, nulla può essere lasciato al caso: ogni azione del sistema intelligente sarà sempre il risultato dell'elaborazione di calcoli volti a verificare i parametri e a determinare le incognite che definiscono le funzioni stesse.

Oggi, i sistemi di *deep learning* trovano applicazione in più settori (v. sez. 2.4) e devono il proprio successo allo sviluppo dei processi delle reti neurali artificiali.

[13]Goh, A. T. C. (1995), "Back-propagation neural networks for modeling complex systems" in *Artificial Intelligence in Engineering*, 9(3), 143-151.

Bisogna anche dire che l'impiego dei sistemi di *deep learning* è la risultante fisiologica di due fattori simbolo dell'era tecnologica che stiamo vivendo: l'aumento esponenziale della mole di dati a disposizione (maggiore è il quantitativo di dati che può analizzare il software maggiore è il livello di apprendimento e quindi la possibilità di avere prestazioni migliori rispetto ad algoritmi più datati) e l'aumento delle performance dei computer che hanno migliorato i risultati ottenibili e ridotto notevolmente i tempi di calcolo[14].

2.3 Rappresentazione della Conoscenza

Alla base delle problematiche legate allo sviluppo di sistemi e programmi di AI vi sono tre parametri che rappresentano i cardini del comportamento umano: una conoscenza non sterile, una coscienza che permetta di prendere decisioni non solo secondo logica e, infine, l'abilità di risolvere problemi in maniera differente a seconda del contesto nel quale ci si trova. L'uso di reti neurali e di algoritmi in grado di riprodurre ragionamenti tipici degli esseri umani nelle differenti situazioni, ha permesso ai sistemi intelligenti di migliorare sempre più le diverse capacità di comportamento. In questo senso, la ricerca si è concentrata sullo sviluppo di algoritmi sempre nuovi che potessero prendere decisioni imitando i diversi comportamenti umani a seconda degli stimoli ambientali. Nel caso degli algoritmi connessi ai sistemi intelligenti dei veicoli, ad esempio, un'automobile senza conducente può decidere, in caso di pericolo, se sterzare o frenare in base alla situazione, ovvero a seconda che le informazioni inviate dai vari sensori permettano di calcolare una maggiore percentuale di sicurezza per il conducente e i passeggeri con una frenata o con una sterzata. Le decisioni prese da un qualunque sistema di AI vengono adottate grazie alla realizzazione di algoritmi che permettono di definire una conoscenza di base ed una conoscenza allargata, ovvero creata tramite l'esperienza.

Per realizzare algoritmi sempre più precisi e complessi, è sorto un vero e proprio settore, definito "rappresentazione della conoscenza", che studia tutte le possibilità di ragionamento dell'uomo e, soprattutto, tutte le possibilità di rendere tale conoscenza comprensibile alle macchine tramite un linguaggio e precisi comandi[15]. Quando si parla di trasferire la conoscenza dell'uomo alla macchina, infatti, non

[14]Govoni L. (2018), "Rete Neurale, Deep Learning e principali applicazioni". Accessibile da: *https://lorenzogovoni.com/deep-learning-e-applicazioni/*.

[15]Rissland, E. L. (1987), "Artificial Intelligence: knowledge representation". Cognitive science: An introduction, second printing.

si parla solo di conoscenza sterile ma anche di esperienza e di possibilità di comprendere nuove informazioni tramite quelle già presenti nel sistema di partenza. Lo studio della rappresentazione della conoscenza è fondamentale per l'AI dato che questa prevede l'utilizzo delle precedenti conoscenze per migliorare e affinare le tecniche di risoluzione dei problemi. Senza un adeguato metodo di memorizzazione della conoscenza molte tecniche di AI sarebbero inutili o molto limitate dato che i sistemi non sarebbero in grado di apprendere dagli stimoli esterni e quindi non potrebbero evolversi. Esistono principalmente due metodologie per fornire le informazioni alla macchina, e quindi rappresentare la conoscenza: la "teoria dei linguaggi formali" e la "teoria delle decisioni". Nel primo caso, quando cioè si utilizza la teoria dei linguaggi formali, si sceglie di utilizzare diversi approcci (quelli riconosciuti sono l'approccio generativo, riconoscitivo, denotazionale, algebrico e trasformazionale) che si rifanno alle teorie delle stringhe e ai loro utilizzi. Le stringhe rappresentano dei veri e propri linguaggi formali le cui proprietà variano a seconda dell'approccio utilizzato. Si può quindi decidere di puntare su un approccio o sull'altro a seconda dei risultati che si intende ottenere, ovvero a seconda del tipo di risposta che si vuole ricevere dalla macchina nelle differenti situazioni. La teoria delle decisioni, invece, si basa su un albero decisionale che permette di valutare per ogni azione le possibili conseguenze; in base a ciò verrà adottata la decisione più conveniente. A seconda delle impostazioni e dello scopo del programma, quindi, il sistema prenderà la decisione che meglio ottimizza il risultato che si vuole ottenere. Va sottolineato che situazioni simili possono prevedere risultati differenti a seconda del tipo di piano di azioni definito dagli algoritmi della macchina. Il metodo degli alberi decisionali è di certo quello maggiormente sfruttato nei sistemi intelligenti utilizzati nel quotidiano. Come funzione un albero di decisione? Basta sapere che si basa su modelli predittivi a partire da una serie di informazioni iniziali e dati di partenza. Tali dati possono poi essere suddivisi in modo tale da definire sia la struttura, ovvero il tipo di previsioni possibili, sia l'accuratezza delle previsioni stesse. A differenziare i sistemi intelligenti non è tanto il numero di dati sul quale si basano le decisioni, ma la precisione nelle predizioni. Difatti, la mole di dati a disposizione per le elaborazioni delle AI può interferire con la precisione del modello utilizzato. Per questo motivo i modelli più accurati presentano un numero di informazioni di partenza spesso inferiore a quello che si può immaginare: la bontà del modello viene assicurata dal tipo e dall'accuratezza dei dati di partenza.

2.4 Campi di applicazione

Per quanto molte persone credano che l'uso di sistemi intelligenti sia relegato a particolari *elite* informatiche, oggi la maturità tecnologica ha fatto sì che l'AI uscisse dall'alveo della ricerca per entrare di fatto nella vita quotidiana. Se come consumatori ne abbiamo importanti assaggi soprattutto grazie a Google, Apple ed Amazon, nel mondo del business la maturità e la disponibilità delle soluzioni tecnologiche ha portato le potenzialità dell'AI in diversi settori.

Ad esempio, nel campo della sanità e dell'*healthcare*, l'AI ha permesso di migliorare molti sistemi tecnologici (per esempio i sistemi vocali sono stati implementati al punto da permettere una comunicazione del tutto naturale anche a chi non è in grado di parlare) ma è sul fronte della diagnosi e della cura che si potranno vedere le sue nuove capacità. Già oggi sono disponibili sul mercato sistemi cognitivi in grado di attingere, analizzare ed apprendere da un bacino infinito di dati ad una velocità inimmaginabile per l'uomo, accelerando processi di diagnosi spesso molto critici o suggerendo percorsi di cura ottimali. Non solo, nelle sale operatorie iniziano a vedersi con maggior frequenza assistenti virtuali a supporto del personale di accoglienza o di chi offre servizi di primo soccorso. Nel settore della gestione del rischio, la prevenzione delle frodi è una delle applicazioni più mature dove l'AI si concretizza con quelli che tecnicamente vengono chiamati "*advanced analytics*", analisi molto sofisticate che correlano dati, eventi, comportamenti ed abitudini per capire in anticipo eventuali attività fraudolente quali, per esempio, la clonazione di carte di credito o l'esecuzione di transazioni non autorizzate. La capacità di analizzare grandissime quantità di dati in tempo reale e di dedurre attraverso correlazioni di eventi, abitudini, comportamenti, attitudini, sistemi e dati di geo-localizzazione e monitoraggio degli spostamenti di cose e persone offre un potenziale enorme anche per il miglioramento dell'efficienza e dell'efficacia della sicurezza pubblica. Nel campo dell'ottimizzazione e gestione della catena di approvvigionamento e distribuzione, l'AI rappresenta un sistema efficace che permette di connettere e monitorare tutta la filiera e tutti gli attori coinvolti. Un caso molto significativo di applicazione dell'AI al settore del *supply chain management* è la gestione degli ordini. In questo caso le tecnologie non solo mirano alla semplificazione dei processi ma anche alla totale integrazione di essi, dagli acquisti fino all'inventario, dal magazzino alle vendite fino ad arrivare addirittura all'integrazione con il marketing per la gestione preventiva delle forniture in funzione delle attività promozionali o della campagne di comunicazione. Un'altra applicazione interessante dell'AI

riguarda l'automazione domestica. Si tratta di tutti quei sistemi che gestiscono gli ambienti in termini di temperatura, illuminazione, sonorità in base alle nostre abitudini e alle nostre preferenze. Termostati come Nest di Google che sono in grado di capire quante persone ci sono una stanza, ma anche reti elettriche che ottimizzano il funzionamento degli elettrodomestici in modo da sfruttare le migliori tariffe energetiche. Oppure tapparelle elettriche collegate al nostro smartphone che si chiudono da sole quando usciamo di casa e si riaprono al nostro ritorno.

Di certo, però, il punto di forza dei sistemi di AI è la capacità di imparare con l'esperienza. I sistemi possono, per esempio, imparare che cosa ci piace guardare, cosa siamo soliti ascoltare o ordinare al ristorante. Una volta scoperti i nostri gusti, gli algoritmi di AI possono darci suggerimenti di consumo e di acquisto in linea con le nostre preferenze. Tali algoritmi di *filtering* sono impiegati da Netflix per consigliare cosa guardare, da Spotify per segnalare la musica e persino da Facebook, il cui algoritmo sceglie quali post mostrarci in base all'esperienza. Anche Google Translate, tramite nuovi sistemi neurali, è in grado di analizzare intere sequenze di frasi fornendo traduzioni vicine alla madrelingua. Ulteriori applicazioni dei sistemi di AI molto note al grande pubblico sono quelle utilizzati su veicoli in grado di guidare senza che vi sia un conducente umano al volante. Si tratta di veicoli ancora in fase sperimentale, ma che raggiungono gradi di sicurezza sempre più elevati soprattutto grazie all'uso di sensori e telecamere in grado di percepire tutto ciò che avviene durante la guida, prendendo decisioni ed effettuando manovre di sicurezza. L'applicazione più interessante, oggetto di un'analisi dettagliata in questo lavoro di tesi, si riscontra a livello di marketing, settore in cui l'AI sta mostrando oggi tutta la sua massima potenza. L'area di impiego maggiore è sicuramente quella della gestione della relazione con gli utenti, e in questo contesto si inseriscono gli assistenti virtuali (chatbot, Siri di Apple, Cortana di Microsoft, Alexa di Amazon) basati essenzialmente su algoritmi di *machine learning*. Tali sistemi sfruttano l'AI sia per il riconoscimento del linguaggio naturale sia per l'apprendimento ed analisi delle abitudini e dei comportamenti degli utenti. In più sono in grado, come vedremo più approfonditamente nella sezione 2.5, di analizzare in tempo reale una grande mole di dati per la comprensione del *sentiment* e delle esigenze del consumatore. L'obiettivo è migliorare la *customer care*, la *user experience*, i servizi di assistenza e supporto ma anche creare e perfezionare sofisticati meccanismi di *engagement* con attività che si spingono fino alla previsione dei comportamenti di acquisto da cui derivare strategie di comunicazione e/o proposta di servizi.

2.5 Virtual Assistant

Nel 1950 Alan Turing pubblicò un articolo dal titolo "*Computing Machinery and Intelligence*", in cui propose un criterio, oggi definito "Test di Turing", in grado di determinare se una macchina è in grado di pensare o meno. Per soddisfare questo criterio un software deve fingere di essere umano in una conversazione in tempo reale in modo che l'interlocutore non sia in grado di distinguere, basandosi solo sul contenuto della conversazione, se stia conversando con un programma o con un essere umano. Questa rappresenta un'importante premessa prima di parlare dell'oggetto di questa sezione: il *virtual assistant* (da ora in poi VA), risultato dell'applicazione di sistemi di AI, nonchè sogno tecnologico che esiste da almeno venti anni e che è stato possibile realizzare soltanto di recente. Ciò che abbiamo visto finora, e in particolare nella sezione 2.2, è il funzionamento tecnologico dei sistemi di AI che, dal punto di vista delle abilità intellettuali, si sostanzia attraverso quattro differenti livelli[16]:

- Comprensione: attraverso la simulazione di capacità cognitive di correlazione tra dati ed eventi, l'AI è in grado di riconoscere testi, immagini, tabelle, video, voci ed estrapolare da questi informazioni.
- Ragionamento: mediante logica, i sistemi riescono a collegare le molteplici informazioni raccolte in modo automatizzato attraverso precisi algoritmi matematici.
- Apprendimento: in questo caso parliamo di sistemi con funzionalità specifiche per l'analisi degli input di dati e per la loro corretta restituzione in output.
- Interazione (*human-machine interaction*): modalità di funzionamento dell'AI in relazione alla sua interazione con l'uomo. È qui che stanno fortemente avanzando i sistemi di NLP che, come discusso nella sezione 1.3, sono tecnologie che consentono all'uomo di interagire con le macchine (e viceversa) sfruttando il linguaggio naturale.

I VA sono agenti software in grado di comprendere i comandi vocali in linguaggio naturale ed eseguire attività richieste dall'utente. Il set di funzionalità comprende l'esecuzione di dettati, la lettura di messaggi di testo, la ricerca, la pianificazione, l'effettuazione di chiamate telefoniche, oltre al fornire tutte quelle informazioni che

[16]Russell, S. J. e Norvig, P. (2016), "Artificial intelligence: a modern approach", Malaysia. Pearson Education Limited.

normalmente verrebbero ricercate in un browser. I VA sono in genere programmi basati sul *cloud* che richiedono il funzionamento di dispositivi e/o applicazioni connessi ad Internet: l'utente rivolge una richiesta che il VA elabora ed archivia nel *cloud* stesso. Poiché l'utente finale interagisce con il VA, la programmazione AI, che richiede enormi quantità di dati, utilizza algoritmi sofisticati per apprendere dai dati di input e migliorare la previsione delle esigenze. In particolare, per abbinare il testo dell'utente o l'input vocale ai comandi eseguibili e avviare un processo di apprendimento, si utilizzano tecniche di NLP, piattaforme di riconoscimento e sintesi vocale, algoritmi di *machine learning* che, tutti insieme, si combinano per garantire la massima flessibilità ed adattabilità agli strumenti utilizzati dal consumatore. Ciò genera una persona virtuale consentendole di parlare e soprattutto di comprendere le domande poste a voce dall'interlocutore umano. Questo nuovo mondo fatto di interfacce conversazionali sta cambiando il modo di interagire con gli altri, con i dispositivi mobili e con i brand. Grazie all'ottimizzazione delle interfacce di conversazione e all'AI, le interazioni con VA e bot saranno sempre più frequenti. I dati parlano chiaro[17]: attualmente 2 ricerche su 10 effettuate da mobile su Google sono vocali e si prevede che entro il 2020 il 50% delle ricerche avverranno senza l'utilizzo dello schermo. A partire dal 2017, le capacità e l'utilizzo dei VA si stanno espandendo rapidamente, con l'ingresso di nuovi prodotti sul mercato. Un sondaggio online a maggio 2017 ha classificato i più utilizzati: Siri di Apple (34%), Assistente Google (19%), Amazon Alexa (6%) e Microsoft Cortana (4%).

Spesso tali VA sono incorporati all'interno di *device* definiti *smart speaker* che, attivati per mezzo di una "*hot-word*", sono in grado di offrire azioni interattive. Esempi sono: Amazon Echo che incorpora Alexa, Apple HomePod che incorpora Siri, Google Home che incorpora Google Assistant, Harman Kardon INVOKE che incorpora Microsoft Cortana. Alcuni *smart speaker* possono anche fungere da dispositivo intelligente che utilizza wi-fi, Bluetooth e altri standard di protocollo *wireless* per estendere l'utilizzo oltre la riproduzione audio (ad esempio per controllare i dispositivi di automazione domestica). Ogni *device* può avere la propria interfaccia e le proprie funzionalità, solitamente avviate o controllate tramite un software applicativo o domotico. Come mostrato nella Figura 2.3, secondo una ricerca di Canalys[18] sono l'accessorio di elettronica di consumo più in crescita con

[17]Gartner (2017), "VPAs in the Enterprise". Accessibile da: https://www.gartner.com/newsroom/id/3790964.

[18]Canalys (2018),"Google beats Amazon to first place in smart speaker market". Accessibile da: *https://www.canalys.com/newsroom/google-beats-amazon-to-first-place-in-smart-speaker-market*.

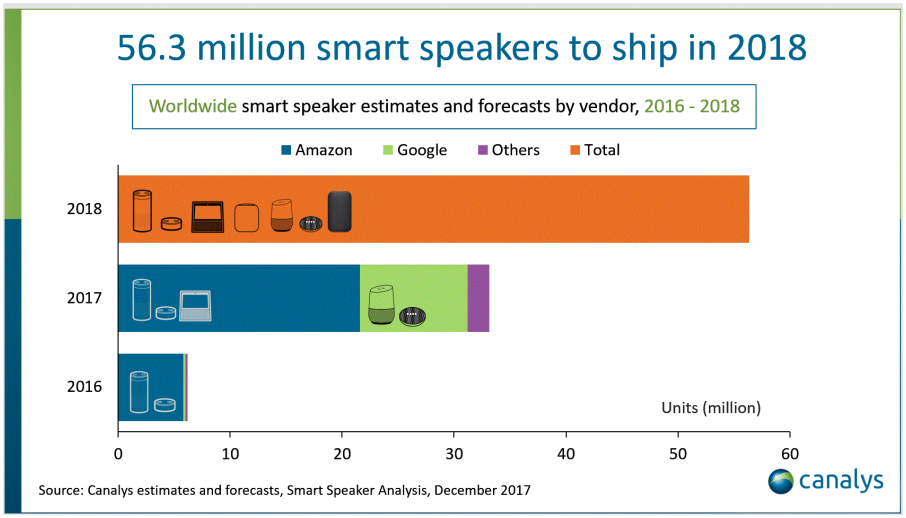

Figura 2.3. Stima di vendita degli smart speaker, 2016-2018

una previsione di vendite di circa 56,3 milioni di unità nel 2018. Sono stati poco meno di 35 milioni nel 2017, furono poco più di 5 milioni nel 2016. Se si prosegue in questa direzione, Amazon e Google, i primi ad arrivare sul mercato rispettivamente con Amazon Echo (lanciato nel novembre del 2014) ed il successivo Google Home, potranno immaginare ulteriori incassi, non necessariamente dalla pura vendita dei dispositivi. I dati del 2017 parlano di poco più di 20 milioni di unità vendute per Amazon e, con un certo ottimismo, della metà per Google. I due cervelli alla base dei dispositivi smart si chiamano Amazon Alexa e Google Assistant e, tecnicamente, rappresentano i software integrati che permettono allo stesso *speaker* di essere intelligente. Sono loro ad ascoltare ed interpretare le richieste dei consumatori, sono loro che, per esempio, completano un acquisto a seguito di un ordine effettuato a voce oltre ad eseguire ogni tipo di attività tipica di un VA: dalla chiamata di un nome in rubrica alla lettura della posta, dalla ricerca sul web alla descrizione di una ricetta. In tutto sono state conteggiate più di 40000 *skill*, numero oggi in piena implementazione. Per quanto concerne le prospettive future, entro il 2020 il 25% delle operazioni di supporto alla clientela andrà ad integrare tecnologie di *chatbot* o di *virtual customer assistant* su tutti i canali di coinvolgimento. Nel 2017 l'impiego di queste tecnologie è stato riscontrato in appena il 2% dei casi. E' la società di analisi Gartner ad elaborare la previsione. A seguito dell'implementazione di soluzioni VA le organizzazioni hanno riscontrato una riduzione del 70% nelle chiamate, chat e/o

richieste via email, oltre ad una crescita del tasso di soddisfazione del cliente e un risparmio del 33% per ogni interazione vocale. Gartner ha scoperto che l'84% delle organizzazioni ha intenzione di incrementare gli investimenti sulle tecnologie di *customer experience* nel corso dei prossimi anni[19].

Gli assistenti virtuali di domani saranno costruiti con tecnologie cognitive più avanzate, che consentiranno a un VA di comprendere ed eseguire richieste multistep ed eseguire attività più complesse[20].

[19]Bai, A. (2018), "Chatbot e assistenti virtuali: nel 2020 svolgeranno il 25% delle operazioni di supporto alla clientela". Accessibile da: *https://pro.hwupgrade.it/news/mercato/chatbot-e-assistenti-virtuali-nel-2020-svolgeranno-il-25-delle-operazioni-di-supporto-alla-clientela_74355.html*.

[20]Rouse, M. (2018), "Virtual Assistant (AI assistant)". Accessibile da: *https://searchcrm.techtarget.com/definition/virtual-assistant*.

Capitolo 3

Metodologia

La classificazione automatica dei documenti testuali per mezzo di algoritmi di *machine learning* conta un grande numero di applicazioni, fornendo soluzioni ai problemi delle aziende che vogliono accedere alle informazioni di proprio interesse in modo semplice, rapido ed intuitivo. L'esigenza deriva dalla crescita esplosiva del web e dal fermento creatosi attorno ai social media, ai siti di recensione e ad ogni strumento portatore della voce del consumatore. Per tale ragione, le aziende, sempre più preoccupate del processo di fidelizzazione del *prospect*, confidano oggi nell'analisi testuale automatizzata volta a monitorare la *web reputation* del brand.

I principali *benefit* portati dall'analisi testuale sono[1]: "l'abilità di monitorare i sentimenti e le opinioni in tempo reale, una chance per ottenere feedback e nuove idee, ottenere aggiornamenti sulla posizione dell'azienda e su quella dei suoi competitor all'interno dell'ambiente competitivo, l'abilità di quantificare le percezioni del mercato". Come visto nelle sezione 1.1, le organizzazioni oggi sono sedute su enormi archivi di dati ma sfortunatamente la maggior parte di questi si presenta sotto forma di *free flow text*. Ciò rappresenta chiaramente un problema enorme in quanto gli algoritmi di *machine learning* preferiscono input ed output ben definiti a lunghezza fissa. Perciò è necessario che il contenuto testuale sia convertito in numeri, nello specifico in vettori numerici. Proponiamo quindi, nella sezione 3.0.1, un metodo per operare, tramite R, il *web scraping* sul sito *amazon.com* con l'obiettivo di estrarre le recensioni del prodotto Amazon Echo e convertirle in metadati da memorizzare ed analizzare all'interno di un database. Tale database, descritto nella sezione 3.0.2, è stato ripulito delle variabili prive di contenuto infomativo (o comunque non strumentali ai fini dell'analisi) ed è stato oggetto di statistiche descrittive

[1]SAS (2018), "SAS Sentiment Analysis". Accessibile da: *https://www.sas.com/it_it/software/sentiment-analysis.html*.

volte a fornire un'idea generale sulla composizione dei dati sui quali operare. Nella sezione 3.1, la raccolta di recensioni è stata convertita prima in corpus e, in seguito ad un *pre-processing* del corpus stesso, in una matrice "Bag of Words". In questa tabella di contingenza, la prima entrata rappresenta la lista dei termini (intesa come vocabolario di tutte le parole uniche che appaiono nell'intera raccolta), la seconda entrata corrisponde al numero di recensione e ogni valore della matrice (cella) indica la frequenza di occorrenza di quel dato termine in quella data recensione. Il valore (textitterm frequency è stato poi utilizzato come *feature* funzionale a due obiettivi di ricerca:

- Proporre, nella sezione 3.2, due metodi di *unsupervised machine learning*: il *Topic Modeling* e gli algoritmi di *clustering*, entrambi volti a scoprire (in modi differenti e confrontabili) gruppi naturali di *item* ed identificare distribuzioni insite nell'insieme dei dati a disposizione.

- Allenare *sentiment text classifier* costruiti tramite metodi di *supervised machine learning*, descritti e confrontati nella sezione 3.4. L'obiettivo è stato classificare una nuova recensione scritta per il prodotto assegnandole un'etichetta di appartenenza ad una categoria: positiva o negativa. In questo modo è possible capire quali sono le regole che definiscono l'appartenenza (o meglio la probabilità di appartenenza) di una recensione ad una o l'altra classe.

La variabile categorica utilizzata come variabile indipendente nei metodi di *supervised machine learning* è stata ottenuta per mezzo di una classificazione della polarità a livello di documento affrontata nella sezione 3.3, in cui l'obiettivo è stato identificare, quantificare e caratterizzare il contenuto sentimentale dell'unità di testo, fornendo un'interpretazione realistica del mercato. Il confronto tra i risultati ottenuti con tre differenti lessici ci ha fornito una prospettiva a più ampio raggio per meglio comprendere l'orientamento emozionale delle recensioni. Quest'analisi ci consentirà di ottenere informazioni generate direttamente dai recensori, fornendo *insight* quantificati sull'*impression* generale che i consumatori hanno circa il prodotto.

3.0.1 Data Scraping

Il web fornisce uno standard semplice ed universale per lo scambio di informazioni e, in quanto tale, deve il suo successo allo sviluppo dell'*Hypertext Markup Language* (HTML) e dell'*HyperText Transfer Protocol* (HTTP). L'HTML rappresenta un mezzo di strutturazione del testo per la presentazione visiva in grado di descrivere la

struttura *intradocument* (il layout ed il format del testo) e la struttura *interdocument* (riferimenti ad altri documenti per mezzo di *hyperlink*). L'HTTP, invece, è un protocollo a livello applicativo usato come principale sistema per la trasmissione di informazioni sul web (ovvero in un'architettura tipica *client-server*). L'introduzione dell'HTTP come standard e l'uso dell'HTML per la composizione dei documenti possono essere considerati come alla radice dell'accettazione universale del web quale mezzo di trasmissione dell'informazione[2]. Oggi l'unità di informazione è tipicamente un file creato da un utente del web e condiviso con altri utenti, rendendo disponibile il proprio nome nella forma di un URL (*Uniform Resource Locator*). Per garantire l'estrazione dell'informazione, però, è necessario che i dati che costituiscono l'informazione stessa siano in forma strutturata. Come visto nella sezione 1.1, il principale problema relativo al web è che è costituito in misura predominante da testo non strutturato. In quest'ottica, l'obiettivo centrale, oggetto di discussione in questa sezione, diviene decomporre l'informazione in unità che possano essere nominate e trasmesse.

La raccolta automatizzata di dati da Internet è vecchia quasi quanto Internet stesso. Sebbene il termine *web scraping* sia di nuova costruzione, la pratica, negli anni passati, già esisteva ed era meglio conosciuta come *screen scraping*, *data mining*, *web harvesting* o variazioni simili. Oggi, il consenso generale sembra favorire il termine *web scraping*, comprendendo un'ampia varietà di tecniche e tecnologie di programmazione, come la *data analysis* e l'*information security*. Il *web scraping* è una pratica di *data mining* che permette di raccogliere dati dal web attraverso qualsiasi mezzo che sia diverso da un programma che interagisce con un'API (*Application Programmable Interface*). Per meglio comprendere, un'API consiste in un insieme di funzioni che permettono di accedere ai servizi di un'applicazione mediante un linguaggio di programmazione[3]. Tutto il lavoro è svolto da un programma automatico (pezzo di codice), detto *scraper*, che opera seguendo quattro step[4]:

1. Invia una *get query* ad uno specifico sito web.
2. Richiede i dati, di solito nella forma di HTML.
3. Analizza il documento HTML per estrarre le informazioni necessarie.
4. Converte i dati in un qualunque formato.

[2]Abiteboul, S. et al. (2000), "Data on the Web: from relations to semistructured data and XML". Morgan Kaufmann.

[3]CCM (2018), "Linguaggi informatici - API". Accessibile da: *https://it.ccm.net/contents/185-linguaggi-informatici-api*.

[4]Upwork (2018), "What is Web Scraping and How Can You Use It?". Accessibile da: *https://www.upwork.com/hiring/for-clients/web-scraping-tutorial/*.

In sintesi, ripercorrendo il processo, il *web scraping* ci consente di estrarre i dati presenti su un sito web in forma non strutturata (HTML *tag*) e convertirli in metadati, memorizzati ed analizzati in locale all'interno di un database. Se si pensa che l'unico modo per accedere ad internet sia attraverso un browser, allora si sta perdendo una vasta gamma di possibilità. Sebbene i browser siano utili per l'esecuzione di JavaScript, la visualizzazione di immagini e la disposizione degli oggetti in un formato più leggibile, i *web scraper* sono eccellenti per la raccolta e l'elaborazione di grandi quantità di dati. Piuttosto che visualizzare una pagina alla volta attraverso la finestra ristretta di un monitor, è possibile visualizzare database che si espandono per migliaia o anche milioni di pagine in una sola volta[5]. In più, i *web scraper* posso raggiungere posti che i tradizionali motori di ricerca non sono in grado di raggiungere. Una tradizionale ricerca Google per "voli più economici per Parigi" si tradurrà in una miriade di pubblicità ed un elenco dei più popolari siti di ricerca voli. Google conosce solo ciò che questi siti dicono nel contenuto delle loro pagine, non i risultati esatti delle varie *query* inserite all'interno di queste applicazioni. Un *web scraper* ben sviluppato può tracciare il costo di un volo nel tempo, attraverso una varietà di siti web, e dirti qual è il momento migliore per acquistare quel biglietto. Le API possono essere fantastiche, se ne trovi una adatta ai tuoi scopi. Possono riguardare diverse tipologie di dati e fornire un flusso conveniente di dati ben formattati da un server all'altro. In generale, è preferibile utilizzare un'API (se ne esiste una), piuttosto che creare un bot per ottenere gli stessi dati. Tuttavia, esistono diversi motivi per cui un'API potrebbe non esistere: se si vogliono raccogliere dati in una raccolta di siti che non dispongono di un'API coesiva; se i dati desiderati sono un insieme finito piuttosto piccolo per cui il *webmaster* non ha pensato nè garantito un'API; se la fonte non ha l'infrastruttura o la capacità tecnica di creare un'API. Anche quando esiste un'API, i dati forniti potrebbero essere insufficienti per i propri scopi. Qui entra in gioco il *web scraping*. Se puoi visualizzare i dati nel tuo browser, puoi accedervi tramite un script di un linguaggio di programmazione; se è possibile accedervi, è possibile memorizzarli in un database e, se puoi salvarli in un database, puoi farci praticamente qualsiasi cosa.

Quasi tutti i linguaggi principali forniscono modi per eseguire lo *scraping* dei dati presenti sul web. Tra i mezzi di estrazione dati, un linguaggio di programmazione comunemente utilizzato è R. Sebbene questo linguaggio, storicamente, non fu progettato con focus sulla manipolazione delle stringhe, oggigiorno offre funzioni

[5]Mitchell, R. (2015), "Web scraping with Python: collecting data from the modern web" in *O'Reilly Media, Inc.*.

idonee alla realizzazione di molteplici *task* nel dominio dei dati testuali. Poichè la sua natura è orientata alla gestione di dati quantitativi, talvolta, alcune sue funzioni rilevanti nell'ambito della *text manipulation* mancano di coerenza. Il recente aumento dell'importanza del *text mining* e del NLP ha, però, favorito un processo (ancora in fieri) di sviluppo ed implementazione di diversi pacchetti in grado di facilitare la manipolazione del testo[6].

Nel caso in analisi, per operare lo *scraping* dei dati dall'HTML della pagina web Amazon del prodotto Echo, è stato installato ed utilizzato il pacchetto rvest, comunemente utilizzato per la manipolazione di codici HTML e XML. Successivamente, è stata applicata una funzione sorgente disponibile su *raw.github.com*[7] per il *parsing* del codice HTML delle pagine Amazon di recensione. Il primo step è stato identificare la *landing page* della pagina recensioni del prodotto ed il relativo ASIN (*Amazon Standard Identification Number*), codice identificativo alfanumerico di 10 caratteri assegnato da Amazon ad ogni articolo[8]. Chiaramente, poichè il prodotto conta migliaia di recensioni (3590), il numero di sottopagine sarà altrettanto rilevante. Facendo clic su una qualsiasi delle sottopagine, lo schema di indirizzamento (URL) cambia. In particolare, sarà composto dall'URL principale più la formula "? page = n", dove *n* è il numero della pagina di recensione (in questo caso qualsiasi numero compreso tra 1 e 2105). Identificato ciò, lo *scraper* deve: definire il numero massimo di pagine cui mandare una *query*, lanciare la *query* a tutte le sottopagine di recensione, operare lo *scraping* delle informazioni da ognuna di queste sottopagine ed infine combinare tutti i dati ottenuti in un unico *data frame*. Si consideri che, in generale, è possibile ispezionare gli elementi visivi di un sito web utilizzando strumenti di sviluppo web nativi per il browser. L'idea alla base è che tutto il contenuto di un sito web, anche se creato in modo dinamico, è codificato in qualche modo nel codice sorgente. Questi *tag* sono in genere sufficienti per individuare i dati che si sta tentando di estrarre. Per ottenere tali dati, abbiamo bisogno di alcune funzioni del pacchetto rvest. Per convertire un sito web in un oggetto XML, si utilizza la funzione read_html(). È necessario fornire un URL di destinazione e la funzione chiama il web server, raccoglie i dati e li analizza. Per estrarre i nodi rilevanti dall'oggetto XML si usa html_nodes(), il cui argomento è il descrittore di classe, anteposto da un punto. L'output sarà una lista di tutti i nodi trovati in quel

[6]Munzert, S. et al. (2014), "Automated data collection with R: A practical guide to web scraping and text mining" in *John Wiley & Sons.*.

[7]Saiko, R. (2016), "Web Scraping and Sentiment Analysis of Amazon Reviews". Accessibile da: *https://justrthings.wordpress.com/2016/08/17/web-scraping-and-sentiment-analysis-of-amazon-reviews/*.

[8]Amazon (2018), "ISBN e ASIN". Accessibile da: *https://www.amazon.it/gp/help/customer/display.html?nodeId=201889580*.

modo, In particolare, è stato avviato un for loop includente quattro funzioni da applicare ad ognuna delle pagine di recensioni:

1. Concatenare l'URL base *"https : //www.amazon.com"* (landing page) con l'ASIN code ed il *page number* utilizzando la funzione paste0().
2. Lettura dell'HTML *code* dell'URL utilizzando la funzione read_html.
3. *Parsing* dell'HTML della pagine utilizzando la funzione amazon_scraper.
4. Combinare il dataset e le recensioni per riga utilizzando la funzione rbind().

```
#parsing data
library(rvest)
ASIN_code = "B06XCM9LJ4"
source("https://raw.githubusercontent.com/rjsaito/Just-R-Things/master/
    Text%20Mining/amazonscraper.R")
dataset=NULL
pages = 2341
for(page_num in 1:pages){
url = paste0("http://www.amazon.com/product-reviews/",ASIN_code,"/?
    pageNumber=", page_num)
doc = read_html(url)
reviews = amazon_scraper(doc, reviewer = F, delay = 2)
dataset = rbind(dataset,reviews)
}
```

Il risultato del processo è stato ottenere un *data frame* composto da 3590 osservazioni (recensioni) per 8 variabili, oggetto di discussione nella prossima sezione.

3.0.2 Dataset

La composizione del *data frame* ottenuto tramite il processo di *web scraping* illustrato nella sezione 3.0.1, è oggetto di descrizione in questa sezione. Il *data frame*, nominato "dataset", consta di 8 categorie informative (variabili) e 3590 righe, dove ogni riga corrisponde ad una recensione ed ogni colonna ad una variabile. Come illustrato nella Tabella 3.1, le categorie informative sono vettori di classe *character* (quindi codici di carattere esprimibili come stringhe di testo) oppure vettori di classe *numeric* (dati quantitativi senza parte frazionaria).

Lo step immediatamente successivo è stato quello di eliminare le colonne, quindi le variabili, prive di contenuto informativo e perciò non strumentali agli obiettivi della ricerca. Dopo il processo di *data cleaning*, il risultato è un *data frame* costituito da due categorie informative, "stars" e "comments", oggetto di studio dell'anali-

	classe	contenuto
title	character	Titolo assegnato dal recensore al proprio commento con il fine di sintetizzarne il contenuto
author	character	Nome dell'autore del commento
date	character	Data di pubblicazione della recensione
ver.purchase	integer	Variabile binomiale che assume valore 0 se l'acquisto risulta verificato, valore 1 in caso contrario. In questo caso il vettore consiste solo di valori 0
comments	character	Testo della recensione
stars	integer	Punteggio assegnato al prodotto dal recensore su una rating scale a 5 punti
format	character	Vettore costituito solo da valori 0
helpful	integer	Vettore contenente il numero di volte che ogni recensione è stata giudicata utile da altri recensori

Tabella 3.1. Categorie informative originarie nel dataset

si. Ora descriviamo queste due variabili fornendo informazioni sulle rispettive distribuzioni di frequenza, indici di posizione ed indici di variabilità. Il vettore stars è costituito da 3590 voti assegnati dai recensori su una *rating scale* a 5 punti. Di questi 3590, il 70% è costituito da punteggi a 5 stelle, il 15% da punteggi a 4 stelle, il 6% da punteggi a 3 stelle, il 3% da punteggi a 2 stelle ed il restante 5% è rappresentato da punteggi ad 1 stella. La distribuzione di frequenze del vettore è mostrata in forma di istogramma nella Figura 3.1. I valori in ascissa rappresentano il numero di stelle assegnate e computato su scala ordinale; il valore in ordinata, invece, corrisponde all numero di recensioni contrassegnate da quel dato punteggio (ovvero la frequenza di occorrenza del dato punteggio). Successivamente sono state calcolate media aritmetica e la deviazione standard del vettore stars, in modo da ottenere una misura del numero medio di stelle assegnate dal recensore al prodotto e quantificare, così, l'ammontare della dispersione intorno al valore medio. Il risultato del calcolo è stato che il *rating* medio è 4.431 mentre la deviazione standard è pari a 1.08. Questo significa che, nella distribuzione, è stata utilizzata mediamente 1 stella in più o meno rispetto al valore medio.

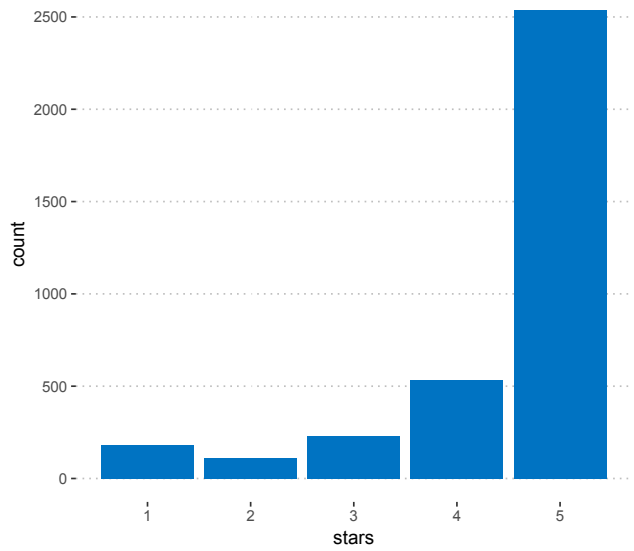

Figura 3.1. Istogramma di frequenze di stars

Il vettore comments, invece, è costituito dalle recensioni scritte da ogni recensore, per un totale di 3590 commenti. A questo punto, come fatto per il vettore stars, si è investigato sulle proprietà descrittive della variabile. Inserendo comments come input della funzione nchar(), si ottiene il numero di caratteri utilizzati per ogni elemento del vettore. In seguito, questo nuovo vettore numerico, nominato "nchar", è stato combinato con il presistente dataset. La tabella a doppia entrata, dopo questo passaggio, è costituita da tre variabili: comments, stars e nchar. Lo step successivo è stato calcolare la media aritmetica e la deviazione standard del vettore nchar, in modo da ottenere una misura del numero medio di caratteri utilizzati per scrivere ogni recensione e quantificare, così, l'ammontare della dispersione intorno al valore medio. Graficando in forma di istogramma i valori assunti dal vettore nchar (v. Fig. 3.2), è possibile farsi un'idea sulla sua distribuzione di frequenze. Il valore in ascissa indica il numero di caratteri utilizzati (per necessità di visualizzazione ho creato dei *bucket* da 25) mentre il valore in ordinata rappresenta il numero di occorrenze. Nel grafico, creato con il pacchetto ggplot2, il gradiente di colore fornisce un'ulteriore chiave di lettura: più è chiaro il colore applicato, maggiore è la frequenza di occorrenza di quel dato valore assunto dalla variabile nchar (come indicato nella legenda). Il risultato del calcolo sugli indici di posizione ha indicato che il numero medio di caratteri utilizzato dai recensori è 145 (intercetta rossa nella Fig. 3.2) mentre la deviazione standard è pari a 271. Questo significa che, nella distribuzione, sono stati utilizzati mediamente 271 caratteri in più o meno rispetto

al valore medio.

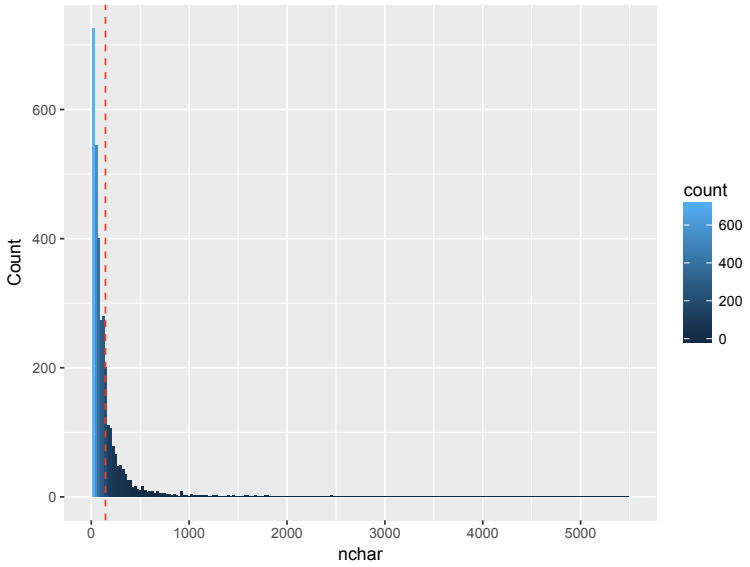

Figura 3.2. Istogramma di frequenze di nchar

La funzione è limitata nell'intervallo [1; 5234]. Come è possibile notare, l'istogramma mostra la presenza di tre *outlier*. Questi dati anomali all'interno dell'insieme di osservazioni suggeriscono che alcuni recensori hanno utilizzato più di 4000 caratteri nella propria recensione. La presenza di tali elementi genera una curva di distribuzione positivamente asimettrica allungando, così, la coda destra. Le statistiche che derivano da popolazioni contenenti *outlier* possono essere fuorvianti; in particolare, la media aritmentica rappresenta uno stimatore sensibile alla presenza di dati anomali. Una stima alternativa più robusta è la c.d. media troncata: si calcola la media aritmetica delle osservazioni dopo aver eliminato i valori più grandi e quelli più piccoli. In questo caso, è più ragionevole adottare il valore mediano come indice sintetico della distribuzione; questo perchè, a differenza della media aritmetica, non soffre la presenza di *outlier*. Nel caso specifico, la mediana è pari a 72 caratteri, valore di gran lunga più basso rispetto alla media precedentemente calcolata (pari a 145).

3.0.3 Correlazione per ranghi di Spearman

La soddisfazione del cliente con i prodotti o i servizi di un'azienda è spesso considerata la chiave del successo e della competitività a lungo termine di un'azienda. Nel contesto del marketing relazionale, la *customer satisfaction* è spesso vista come

un fattore determinante per la fidelizzazione dei clienti[9] In questa prospettiva, si inserisce un nuovo spunto sul quale potrebbe essere interessante investigare: la significatività dell'associazione tra due variabili del nostro dataset, ovvero il numero di caratteri utilizzato dai recensori per scrivere il proprio commento ed il numero di stelle assegnato per il prodotto. Il fine è verificare se la soddisfazione post-acquisto del cliente è direttamente proporzionale al numero di caratteri utilizzati dallo stesso per recensire il prodotto. In questa sezione vedremo come realizzare tale valutazione utilizzando come mezzo di verifica d'ipotesi il coefficiente di correlazione per ranghi di Spearman, sovente indicato con ρ.

Prima di introdurre il coefficiente di correlazione ρ e la sua significatività nel test d'ipotesi, per comprendere a fondo i motivi della scelta di tale metodo ed il suo significato, è necessaria una breve introduzione sul concetto di correlazione tra variabili statistiche. La correlazione è un'analisi bivariata che misura la forza dell'associazione tra due variabili e la direzione di tale relazione. In termini di forza dell'associazione, il valore che il coefficiente può assumere varia tra -1 e +1 (estremi inclusi): un valore estremo di ± 1 indica un grado di perfetta associazione tra le variabili; un valore pari (o prossimo) a 0 indica, invece, indipendenza tra le stesse[10]. Per meglio comprendere l'intensità della forza della correlazione tra due variabili, di solito, si assume come guida la classificazione di intervalli numerici con relativa categorizzazione verbale illustrata nella Tabella 3.2.

range di valori	**livello di correlazione**
.00 - .19	correlazione molto debole
.20 - .39	correlazione debole
.40 - .59	correlazione moderata
.60 - .79	correlazine forte
.80 - 1.0	correlazione molto forte

Tabella 3.2. Livelli di correlazione

La direzione della relazione, invece, è indicata dal segno del coefficiente: segno positivo sta per relazione positiva, segno negativo sta per relazione negativa. Questo

[9] Hennig, T. et al. (1997), "The impact of customer satisfaction and relationship quality on customer retention: A critical reassessment and model development" in *Psychology & marketing*, 14(8), 737-764.

[10] Statstutor (2018), "Spearman's correlation". Accessibile da: *http://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf*.

vuol dire che, nel caso in cui la relazione risulti positiva, all'aumentare del valore di una delle due variabili corrisponderà un incremento nel valore dell'altra variabile (o viceversa); se la relazione è negativa, invece, all'aumentare del valore di una variabile, diminuirà il valore dell'altra (o viceversa). Come visto precedentemente, la misura dell'incremento/decremento è data dal valore numerico assunto dal coefficiente. In generale, la correlazione tra variabili può essere misurata con l'uso di diversi coefficienti. I tre più popolari sono: il coefficiente di Pearson (per le scale a intervalli o rapporti equivalenti), il coefficiente ρ di Spearman (per variabili di tipo ordinale) e il τ di Kendall (alternativo al ρ).

Il coefficiente ρ è un indice di correlazione non parametrico che permette di valutare la forza del rapporto tra due variabili quando le assunzioni per il modello di correlazione parametrica non possono essere soddisfatte. Le ipotesi di tale correlazione per il test di associazione prevedono che i dati di almeno una delle due variabili siano ordinali (l'altra può essere ordinale o di misura) e che i punteggi di una variabile siano monotonicamente correlati all'altra variabile. Ciò presuppone che, a differenza del coefficiente di Pearson, non vi sia alcun obbligo di normalità; ecco perchè il ρ rappresenta una statistica non parametrica [11]. Infatti, a differenza del coefficiente di correlazione *product-moment* di Pearson, il coefficiente ρ non richiede alcuna assunzione di linearità nella relazione tra le variabili, né richiede che queste siano misurate su scale intervallari. In sintesi, il coefficiente di Spearman non può essere inteso come una misura della relazione lineare tra due variabili in quanto valuta semplicemente come la relazione tra queste possa essere descritta tramite una funzione monotona, senza fare alcuna ipotesi sulle distribuzioni di frequenza. Il valore del coefficiente ρ è dato dalla seguente formula:

$$\rho = 1 - \frac{6 \sum d_i^2}{n(n^2 - 1)} \tag{3.1}$$

dove

d = distanze a coppie dei ranghi delle variabili x_i e y_i.

n = numero complessivo di osservazioni.

Nel caso in esame, il vettore stars consiste di valori compresi nell'intervallo [1; 5] che, come sappiamo, rappresentano i punteggi assegnati dai recensori al prodotto. Poichè tali valori sono numeri interi positivi misurati a livello di scala ordinale, per calcolare la forza e la direzione dell'associazione tra il vettore stars ed il numero di caratteri utilizzati per recensire l'acquisto (vettore nchar), ha più senso applicare

[11] Hauke, J. e Kossowski, T. (2011), "Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data" in *Quaestiones geographicae*, 30(2), 87-93.

un metodo che utilizzi i ranghi piuttosto che fare assunzioni sulle distribuzioni. In quest'ottica, il calcolo della correlazione per mezzo del coefficiente di Spearman si configura come il metodo più logico ed efficiente. Calcolando la correlazione, il test di verifica d'ipotesi restituisce un valore del ρ compreso nell'intervallo [-1; +1] che, come già visto, indica, nel segno e nel valore, il tipo e la forza della correlazione. Il coefficiente, quindi, serve a verificare o rigettare l'ipotesi nulla di indipendenza tra le variabili[12]. Di seguito l'output del test di verifica d'ipotesi.

```
cor.test(dataset$nchar,dataset$stars,method="spearman")

Spearman's rank correlation rho
data:  dataset$nchar and dataset$stars
S = 9821800000, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
       rho
-0.2736798
```

Il test di verifica d'ipotesi ha restituito un valore del coefficiente ρ pari a -0.2736798, indicativo di una correlazione debole tra i vettori nchar e stars (v. Tab. 3.2). Stando al segno del coefficiente, mentre il valore del vettore nchar tende ad incrementare, il valore di stars tende leggermente a diminuire. Ad un livello di confidenza di .95, il p-value risulta $< .05$, quindi statisticamente significativo. Questo indica che possiamo rigettare l'ipotesi nulla di indipendenza tra le due variabili e che quindi le stesse sono correlate negativamente in modo significativo. Come mostrato nella Figura 3.3, la correlazione tra le due variabili è molto debole. In qualunque caso, il test suggerisce che, probabilmente, maggiore è la soddisfazione post-acquisto nei confronti del prodotto (e quindi il numero di stelle assegnato) più i recensori tendono a dedicare meno parole nel recensire l'acquisto fatto.

Si tenga a mente che la correlazione non include il concetto di causa-effetto ma solo quello di rapporto tra variabili. In questo senso, la correlazione ci permette di affermare che tra due variabili c'è una relazione sistematica, ma non che una causi l'altra.

[12]Laboratorio statitica (2018), "Il coefficiente di correlazione di Spearman per ranghi". Accessibile da: *https:laboratoriostatistica.files.wordpress.com/2014/09/spearman.pdf*.

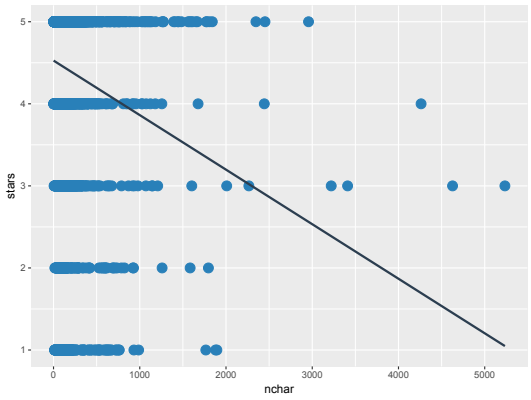

Figura 3.3. Correlazione tra stars e nchar

3.1 Corpus e Modello Bag of Words

La classificazione automatica di documenti testuali all'interno di un set predefinito di categorie è un campo che conta un grande numero di applicazioni, fornendo soluzioni ai problemi delle aziende (e del consumatore) che vogliono accedere alle informazioni di proprio interesse in modo semplice, rapido ed intuitivo. La principale complicazione relativa ai documenti testuali è la mancanza di struttura. Il problema sta nel fatto che gli algoritmi di *machine learning*, per lavorare, preferiscono input ed output ben definiti a lunghezza fissa. Tali tecniche, infatti, non possono funzionare se applicate direttamente su un testo non elaborato; è necessario che il contenuto di classe caratteriale sia convertito in numeri (nello specifico in vettori numerici).

"*Nell'elaborazione del linguaggio, gli x vettori devono derivare da dati testuali, al fine di riflettere le varie proprietà linguistiche del testo*"[13]".

Il processo cui Goldberg fa riferimento è definito "estrazione di caratteristiche", altrimenti detto "codifica delle caratteristiche". In questo contesto si inserisce il *Bag of Words* (da ora in poi BoW), modello di analisi e rappresentazione testuale in grado di estrarre date *feature* da documenti testuali ed inserirle prima in un *corpus* testuale, poi all'interno di una matrice a due dimensioni. Con la creazione della matrice BoW, l'obiettivo è trasformare ogni recensione in un vettore impiegabile come dato di input (o output) per algoritmi di *machine learning*. Il processo di creazione consta fondamentalmente di due fasi:

[13]Goldberg, Y. (2017), "Neural network methods for natural language processing" in *Synthesis Lectures on Human Language Technologies*, 10(1), 1-309.

1. Date *n* recensioni, alla prima si applica il metodo del punteggio binario, ovvero si contrassegna l'occorrenza di una parola come valore booleano: 0 se assente, 1 se occorre. In questo modo, la recensione viene convertita in un vettore di punteggi e le parole che la costituiscono rappresentano il vocabolario in forma embrionale.

2. Il processo di trasformazione vettoriale di cui al punto 1 viene iterato per tutte le recensioni fino ad ottenere una matrice in cui un'entrata corrisponde alla lista dei termini (intesa come vocabolario di tutte le parole uniche che appaiono nell'intera raccolta), mentre l'altra entrata indica la frequenza di occorrenza di ogni termine per ogni recensione.

Si noti come, seguendo tale logica, si perde ogni informazione sull'ordine, struttura del documento e relazione semantica tra le parole. Al contempo, però, il contenuto informativo rimane intatto. Per la creazione di un vocabolario funzionale all'estrazione di *feature* dalle recensioni, ciò che interessa è unicamente l'insieme dei termini che occorrono nella raccolta di recensioni.

"*In questo approccio, guardiamo all'istogramma delle parole all'interno del testo, cioè considerando il conteggio di ogni parola come caratteristica*".

Il modello, infatti, si preoccupa solo di conteggiare il numero di volte che ogni parola figura all'interno delle recensioni, non sul dove il termine sia locato. Questo perchè è la frequenza di occorrenze delle parole ad essere utilizzata come *feature* unica per allenare i *text classifier*, di cui parleremo nelle sezione 3.4. Con l'aumentare delle dimensioni del vocabolario, aumenta anche la rappresentazione vettoriale dei documenti. Potete immaginare come, per un *corpus* costituito da migliaia di recensioni, la lunghezza del vettore sarebbe altrettanto lunga, contando un numero rilevante di posizioni. Al contempo, potrebbe accadere che ogni documento contenga pochissime delle parole che costituiscono il vocabolario totale; ciò si tradurrebbe in un "vettore sparso" (o "rappresentazione sparsa"), ovvero in un vettore con un numero enorme di punteggi pari a 0. I vettori sparsi richiedono più memoria e risorse computazionali e, durante il processo di *modeling*, un numero iperbolico di posizioni (o dimensioni) renderebbe tale processo troppo impegnativo per i tradizionali algoritmi di *machine learning*. Per questo motivo, quando si utilizza

l'approccio BoW, persiste una logica pressione alla diminuizione della dimensione del vocabolario. Tra le opportunità di riduzione del vocabolario figurano[14]:

- Rimozione della punteggiatura e dei caratteri numerici.
- Rimozione delle *stopword*.
- Processo di *stemming*.

Analizziamo ora i punti annoverati. In riferimento al primo punto, è possibile asserire come la punteggiatura rappresenti spesso un buon indicatore del *sentiment* in quanto ci permette di dedurre molteplici informazioni. In genere, però, tende ad ingombrare la classificazione, frammentando i *token* inutilmente. Per questo motivo, molti *tokenizer* sono orientati verso l'eliminazione della punteggiatura. Lo stesso ragionamento è traslabile nei confronti dei caratteri numerici. Tale logica di rimozione assume senso perchè il *pre-processing* rappresenta, in *primis*, un tentativo di generalizzare i dati rimuovendo ciò che risulta inutile ai fini della ricerca. Il secondo problema è quello delle *stopword*. Per l'assegnazione dei punteggi relativi ad ogni parola, esistono metodi che conteggiano il numero di volte che un termine appare in un documento e calcolano, all'interno di ciascun documento, la frequenza relativa con cui un dato vocabolo appare rispetto a tutte le altre parole. Una complicazione inerente questa valutazione è che, in una data recensione, potrebbero figurare parole che dominano in termini di occorrenza (assumendo un punteggio totale molto alto), ma che, al contempo, risultano prive di contenuto informativo in relazione al modello. Altri termini invece più rari potrebbero rappresentare parole specifiche del dominio. Un approccio risolutivo consiste nel ridimensionare la frequenza delle parole in modo che i punteggi relativi a termini frequenti ma privi di contenuto informativo siano penalizzati. In particolare, la fase di *pre-processing* prevede l'eliminazione delle parole di uso comune, sovente dette *stopword*. Il terzo problema, infine, è quello relativo al processo di *stemming*. Per motivi grammaticali, all'interno dei documenti testuali vengono utilizzate diverse forme di una parola (vedi per esempio *organize*, *organizes* e *organizing*). In più, ci sono famiglie di parole derivate correlate con significati simili (del tipo *democracy*, *democratic* e *democratization*). Ai fini dell'applicazione di algoritmi di analisi testuale, risulta utile ridurre le forme flesse e le forme derivate correlate di una parola ad una

[14]Traspinar, A. (2015), "Text Classification and Sentiment Analysis". Accessibile da: *http://ataspinar.com/2015/11/16/text-classification-and-sentiment-analysis/*.

forma base comune[15] (radice). In questo contesto si inserisce l'attività di *stemming*. Come mostrato nella Figura 3.4, questo metodo di *Information Retrieval* identifica la radice della parola per generalizzare le operazioni di interrogazione e selezione dei documenti e, all'interno del *corpus*, tutti i termini vengono sostituiti con le relative radici. Il risultato finale è una versione del testo con la stessa quantità di termini ma con meno varianti. Nello *stemming* la radice non deve essere identica a quella morfologica della parola; di solito è sufficiente che le parole correlate siano mappate sulla stessa radice, anche se questa non è di per sè una radice valida.

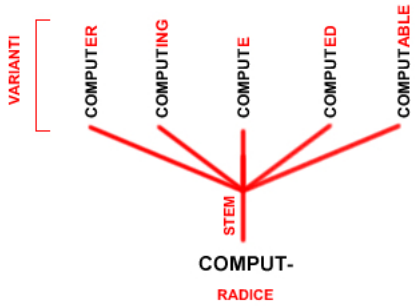

Figura 3.4. Processo di stemming

Lo stesso metodo viene applicato alle parole di interrogazione dell'utente (*query*); in un sistema IRS (*Information Retrieval Stemming*), le *query* vengono ridotte alla loro radice. In questo modo, il processo di ricerca e di *matching* delle informazioni è più efficace, in quanto la radice espande la selezione su più documenti. Di solito, per ridurre la complessità del testo, le *stopword* vengono eliminate prima dello *stemming*, in modo da minimizzare il numero di termini da ridurre alla radice. Così facendo diminuisce il tempo di esecuzione ed aumenta l'efficienza computazionale, quindi si semplifica l'algoritmo. Nello *stemming*, la riduzione a radice può avvenire con diversi gradi di *stem*. Una scelta equilibrata è senza dubbio migliore e più efficace rispetto a quelle estreme. Difatti, aumentando il grado di *stem* la radice diventa più corta e si amplia il numero di termini. La conseguenza fisiologica è l'*over-stemming*: si individuano più legami tra le parole con lo stesso significato ma si rischia di includere anche quelle con significato diverso; in questo caso, lo svantaggio principale è la perdita di specificità. Viceversa, riducendo il grado di *stem* aumenta la lunghezza della radice e si riduce la quantità di parole considerate nella famiglia dei termini. La conseguenza è l'*under-stemming*: i termini sono più

[15]Cambridge University (2009), "Stemming and lemmatization". Accessibile da *https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html*.

specifici ma si rischia di perdere qualche relazione tra le parole che hanno lo stesso significato; lo svantaggio è la perdita di generalità. Nell'ambito del *text mining* esistono tre differenti algoritmi di derivazione: lo *stemmer* di Porter, lo *stemmer* di Lancaster e lo *stemmer* WordNet. In generale, quello che ci aspettiamo da un buono *stemmer* è che riconosca, con la maggiore accuratezza possibile, la correlazione tra termini cui attribuiamo una semantica comune, e sfrutti questa correlazione per sostituire tutti i termini correlati con il tema corrispondente[16].

Vediamo ora come applicare i principi di *text mining* finora discussi sulla raccolta di recensioni di cui disponiamo. Partendo dai dati testuali contenuti all'interno del corpo di recensioni, alcune funzioni del pacchetto tm e SnowballC consentono di creare un *text corpus* ed ottenere da questo una matrice BoW. Come primo passaggio, applicando la funzione VCorpus(), è stato creato un *corpus* di testo costituito dalle 3590 recensioni. La funzione tm_map() ha permesso, in seconda battuta, di pre-processarlo. In particolare ha consentito di:

- Convertire il testo in carattere minuscolo. Uniformare i caratteri è utile per evitare che l'algoritmo consideri come elementi differenti termini del tipo: "echo" - "Echo".
- Rimuovere la punteggiatura ed i numeri per evitare che l'algoritmo consideri come elementi differenti termini del tipo: "echo" - "echo1" - "@echo".
- Eliminare dal corpo testuale le *stopword*. L'obiettivo è stato rimuovere tutte le parole comuni di testo che non avevano a che fare con uno specifico tema trattato nelle recensioni, quindi non significative per l'analisi testuale.
- Attivare il processo di *stemming* in modo da minimizzare la grandezza del dizionario ad una misura efficiente. Poichè all'interno della matrice BoW interessa l'unicità del termine, lo *stemming* consente di riportare i termini solo in base alla radice e non in base alla desinenza eliminando, così, il problema di inutili *multiple counting*.

```
1 #corpus
2 library(tm)
3 library(SnowballC)
4 corpus = VCorpus(VectorSource(dataset$comments))
5 corpus = tm_map(corpus, content_transformer(tolower))
6 corpus = tm_map(corpus, removePunctuation)
```

[16]Porter, M. F. (2001) "Snowball: A language for stemming algorithms".

```
corpus = tm_map(corpus, removeNumbers)
corpus = tm_map(corpus, removeWords, stopwords("english"))
```

La Figura 3.5 mostra, in forma di *wordcloud*, le parole più frequenti contenute nel *corpus* in seguito alla fase di *pre-processing* dello stesso. Questa modalità di rappresentazione consiste in una lista pesata che dà maggior rilevanza alle *keyword*: il peso dei *tag*, reso con caratteri di dimensioni diverse, è inteso esclusivamente come frequenza di occorrenza del termine all'interno delle recensioni: più è grande il carattere, maggiore è il numero di apparizioni della data *keyword* nel corpo testuale.

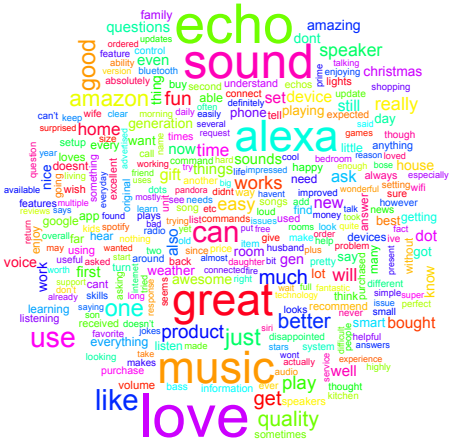

Figura 3.5. Wordcloud

Ora che il *text corpus* è stato pre-processato, è possibile estrarre la frequenza di ogni singolo termine. Per fare questo, il *corpus* è stato convertito in una matrice, nominata "tdm", utilizzando la funzione TermDocumentMatrix().

```
tdm = TermDocumentMatrix(corpus)
```

La matrice tdm contiene 3590 documenti (recensioni) e 4252 distinti termini con il 100% di sparsity. Questo livello di *sparsity* suggerisce che, all'interno della raccolta di recensioni, figura una grande mole di termini che appare raramente; quindi non tutte le parole hanno la stessa rilevanza. È bene sottolineare come, in realtà,

questo indice di dispersione sia giustificato dalla presenza di un enorme numero di dati testuali. Per valutare la rilevanza dei vari termini contenuti nelle recensioni è possibile adottare come indicatore il *weighting value*, metrica che misura la frequenza di occorrenza di ogni parola in ogni singola recensione e che quindi ne pondera il peso all'interno della raccolta. Nella matrice tdm le colonne corrispondono alla lista delle recensioni, ogni riga rappresenta uno dei termini contenuti nelle recensioni mentre ogni cella (valore della matrice) contiene il numero di apparizioni di quel dato termine in quel dato documento. Ora che abbiamo costruito la matrice BoW, la stessa è pronta per essere utilizzata come base per l'applicazione di algoritmi di *machine learning*. L'approccio BoW si configura come un modello facilmente implementabile ed in grado di offrire grande flessibilità per la personalizzazione dei dati di testo. È questo il motivo per cui ha riscontrato grande successo nei problemi di *language modeling* e classificazione dei documenti. Anche se largamente impiegato per la risoluzione di tali problemi, l'approccio presenta comunque alcune carenze:

- Il vocabolario richiede un'attenta progettazione, in particolare per gestire la dimensione che influisce sulla scarsità delle rappresentazioni del documento.

- Problema della *text sparsity*: le rappresentazioni sparse sono più difficili da modellare sia per ragioni computazionali (spazio e complessità temporale) sia per ragioni di carattere informativo. La sfida è far sì che i modelli sfruttino poche informazioni in uno spazio rappresentativo piuttosto ampio.

- Scartare l'ordine delle parole equivale ad ignorare il contesto, quindi il significato delle parole nel documento (semantica).

In qualunque caso, il BoW, poichè in grado di catturare informazioni sufficienti per fare predizioni accurate, si prefigura come rappresentazione standard del testo per lo sviluppo di classificatori lineari di *machine learning*, spesso preferiti a modelli molto più complessi per la loro efficienza, robustezza ed interpretabilità[17]. In quest'ottica, come vedremo nelle sezioni seguenti, la frequenza di occorrenza delle singole parole ottenuta con il modello, verrà utilizzata come *feature* per allenare tali *text classifier*, penultimo step dell'intero processo illustrato nella Figura 3.6.

[17]Porcu, V. (2016), "Introduzione al machine learning con R".

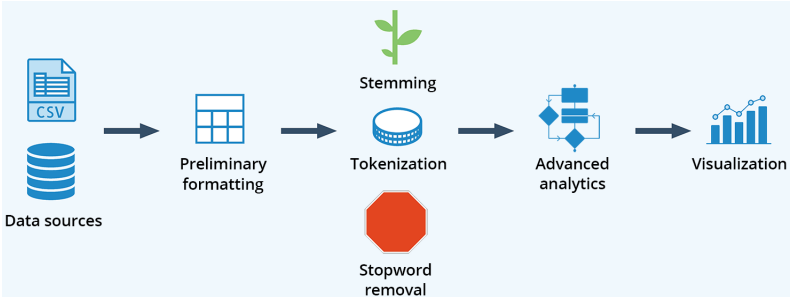

Figura 3.6. Processo di opinion mining

3.2 Algoritmi di Unsupervised Machine Learning

Ogni giorno vengono raccolte, via web, grandi quantità di dati non strutturati. Man mano che diventa disponibile una mole sempre maggiore di informazioni, risulta anche più difficile accedere a ciò che stiamo cercando. Per questo motivo, nel caso in cui si disponga di una rilevante collezione di dati testuali (come nel nostro caso di recensioni Amazon), è necessario adottare un metodo di organizzazione e classificazione testuale ci che aiuti a decifrare e comprendere le informazioni ivi contenute. In questa sezione illustrerò come ho applicato due tecniche di *unsupervised machine learning*, il Topic Modeling Method e gli algoritmi clustering, entrambe in grado di creare gruppi di recensioni in base ad un dato criterio di classificazione.

Il Topic Modeling Method, discusso nella sezione 3.2.1, fornisce metodi per organizzare, comprendere e sintetizzare grandi raccolte di informazioni testuali. In particolare, l'algoritmo permette di scoprire topic latenti all'interno di una collezione di documenti, associare questi ultimi ai topic individuati ed infine utilizzare l'associazione calcolata per organizzare ed interpretare i dati testuali. Tra gli approcci più frequentemente utilizzati figura il Latent Dirichlet Allocation, tecnica in grado di mostrare quanto di ogni topic è presente in ogni singolo documento tramite distribuzioni multinominali con probabilità di appartenenza associate[18]. In questo senso, il Topic Modeling, partendo da una raccolta di documenti, cerca quei gruppi di parole in grado di rappresentare al meglio le informazioni contenute. L'algoritmo, in alternativa, può essere pensato come una forma di *text mining* in grado di identificare *pattern* ricorrenti di parole. Anche il clustering, come il Topic Modeling, è un'applicazione di *data mining* utilizzata per scoprire gruppi naturali di *item* ed identificare distribuzioni insite nell'insieme dei dati a disposizione. Questa

[18] Blei, D.M. et al. (2003), "Latent dirichlet allocation" in *Journal of machine Learning research*, 3(Jan), 993-1022.

tecnica di analisi multivariata consente di creare insiemi di dati (cluster) basandosi su misure relative alla somiglianza (distanza reciproca) tra le unità statistiche. In base a tale principio, l'appartenenza o meno ad un cluster dipende da quanto l'elemento preso in esame risulta distante dal cluster stesso. Secondo questo approccio, oggetti tra loro "simili" apparterranno allo stesso gruppo, garantendo, così, minima devianza *within-groups* e massima devianza *between-groups*[19]. Bisogna sottolineare, comunque, che la bontà delle analisi ottenute dagli algoritmi di clustering dipende molto dalla scelta della metrica, e quindi da come viene calcolata la distanza. Gli algoritmi applicati in questo lavoro di tesi sono stati quello di clustering gerarchico agglomerativo e quello delle *k*-medie proposto da MacQueen[20] nel 1967. Il primo organizza i dati in sequenze nidificate di gruppi rappresentabili in una struttura ad albero; il secondo, invece, è un algoritmo partizionale che opera determinando il partizionamento dei dati in cluster in modo da ridurre il più possibile la dispersione all'interno del singolo cluster e, al contempo, in modo da aumentare la dispersione tra i cluster[21]. La differenza tra il Topic Modeling e gli algoritmi di clustering si fonda sostanzialmente sull'approccio impiegato: nel primo, ogni topic è definito da un cluster di parole (distribuzione multinominale) e relative probabilità di appartenenza al dato topic; nel clustering, invece, le recensioni sono raggruppate in differenti sottoinsiemi in base alla frequenza di occorrenza del termine, adottata come misura di similarità. Poichè il metodo del Topic Modeling genera una rappresentazione dei documenti in un *topic space*, l'idea di base è utilizzare tale rappresentazione come base per performare, in seconda battuta, gli algoritmi di clustering. L'obiettivo è stato verificare se tali algoritmi mi avrebbero restituito un'associazione tra recensione e topic in modo similare a quanto ottenuto con il metodo del Topic Modeling. In definitiva, queste tecniche di *unsupervised machine learning* di effettuare operazioni di segmentazione sui dati, cioè di individuare gruppi omogenei, o tipologie (in questo caso topic), che presentano delle regolarità al loro interno in grado di caratterizzarli e differenziarli dagli altri gruppi.

[19]Jain, A.K. et al. (1999), "Data clustering: a review" in *ACM computing surveys*(CSUR), 31(3), 264-323.

[20]MacQueen, J. (1967), "Some methods for classification and analysis of multivariate observations" *in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, No. 14, pp. 281-297).

[21]Data skills (2018), "Tecniche di clustering". Accessibile da: *https://www.dataskills.it/tecniche-di-clustering/*.

3.2.1 Topic Modeling Method

Nell'era dell'informazione che stiamo vivendo, l'ammontare di materiale interattivo scritto (non strutturato) con cui ci interfacciamo ogni giorno risulta semplicemente aldilà delle nostre capacità di processamento. In quest'ottica, i *topic model* possono aiutarci a decifrare i dati testuali, interpretarli ed organizzarli, offrendo *consumer insight* rilevanti per le aziende e, più in generale, per tutti gli attori operanti sul mercato. In questa sezione vedremo come applicare, su una raccolta testuale, il Topic Modeling Method, uno strumento di *text mining* che, basandosi sull'approccio Latent Dirichlet Allocation, è in grado di scoprire ed analizzare strutture semantiche latenti all'interno di un qualsiasi *text body*.

L'obiettivo annunciato e perseguito è raggruppare le recensioni in base al contenuto (ovvero in base al topic discusso) in modo da offrire un criterio di classificazione. Con questa premessa, il Topic Modeling si è rivela in grado di fornire mezzi efficaci per la scoperta e la sintesi automatica delle opinioni a partire da una mole rilevante di *user-generated data*. Tipicamente, un documento tratta più topic in differenti proporzioni: se, per esempio, una delle recensioni di Amazon Echo trattasse per l'80% del prezzo del prodotto e per il 20% del funzionamento dello stesso, allora, probabilmente, all'interno della recensione mi aspetterei di trovare otto volte più parole relative al prezzo piuttosto che al funzionamento. Il modello del Topic Modeling cattura questa intuizione in un *framework* matematico che consente di esaminare un set di documenti e scoprire, in base alle statistiche relative alle parole di ciascuna delle recensioni, quali potrebbero essere i topic trattati e quale il bilanciamento degli stessi all'interno del *text document*. Tra i mezzi forniti dal Topic Modeling figura il Latent Dirichlet Allocation (da ora in poi LDA), una tecnica di *unsupervised machine learning* in grado di identificare gruppi latenti e naturali di *item* all'interno di una collezione di documenti testuali[22]. Utilizzando l'approccio BoW, il LDA tratta ogni documento come un vettore di *word count*. In particolare, ogni recensione è rappresentata come una distribuzione di probabilità sui topic; i topic, invece, sono rappresentati come distribuzioni di probabilità sui termini che occorrono [23]. Ogni recensione, in pratica, viene mostrata come un mix dei topic presenti nel *corpus*. Questo perchè l'algoritmo propone che ogni parola del documento sia attribuibile ad uno dei topic discussi. ll campionamento compresso

[22]Blei, D.M. et al. (2003), "Latent dirichlet allocation" in *Journal of machine Learning research*, 3(Jan), 993-1022.

[23]Hong, L. E Davison, B. D. (2010), "Empirical study of topic modeling in twitter" in *Proceedings of the first workshop on social media analytics* (pp. 80-88). ACM.

di Gibbs è un modo in cui il LDA apprende gli argomenti e le rappresentazioni di ciascun documento. La procedura è la seguente:

1. L'algoritmo passa attraverso ogni documento e assegna casualmente ogni parola del documento a uno dei k topic (con k scelto a priori). Tale assegnazione randomica fornisce una rappresentazione di tutti i documenti e delle distribuzioni di parole di ognuno dei topic.
2. Per migliorare tale assegnazione, l'algoritmo, per ogni documento d, passa attraverso ogni parola w e calcola:
   - p (topic t | documento d), ovvero la percentuale di parole nel documento d che sono assegnate al topic t.
   - p (parola w | topic t), ovvero la percentuale di assegnazioni al topic t su tutti i documenti d provenienti dalla parola w.
3. Riassegna la parola w ad un nuovo topic t', dove scegliamo l'argomento t' con probabilità p (topic t' | documento d) * p (parola w | topic t'). Questo modello generativo predice la probabilità che l'argomento t abbia generato la parola w.
4. Ripetendo l'ultimo passaggio più volte, si raggiunge uno stato stazionario in cui le assegnazioni dei topic possono definirsi piuttosto buone.

In sintesi, questa tecnica è simile al clustering overlapping applicato sui dati numerici: le recensioni sono trattate come un mix di topic, dove ogni topic è rappresentato come una distribuzione multinominale, ovvero come una distribuzione di probabilità sulle parole. Seguendo questi passi, ora vedremo come il Topic Modeling Method è stato applicato sul dataset e quali sono stati i risultati ottenuti utilizzando i principi di tidytext. A monte, è necessaria l'installazione di specifici pacchetti, ovvero tidytext, topicmodels e dplyr, le cui funzioni permettono di performare l'algoritmo. Il primo step è stato creare una matrice, nominata "dtm", utilizzando la funzione DocumentTermMatrix(). La matrice risultante rappresenta la trasposta della matrice tdm già incontrata nella sezione 3.1. In questa matrice ogni colonna rappresenta uno dei termini contenuti nelle recensioni, le righe corrispondono alla lista delle recensioni mentre ogni singolo valore della matrice (cella) contiene il numero di apparizioni di quel dato termine in quel dato documento. Nel complesso la matrice contiene 3590 documenti e 4252 distinti termini con il 100% di *sparsity*. Questo dato di *sparsity*, come già visto nella sezione 3.1, è indicativo di una grande dispersione testuale. Poichè all'interno della raccolta di recensioni molti termini appaiono sporadicamente, allora non tutti avranno la stessa rilevanza; tendenzialmente, maggiore è la frequenza di occorrenza di un termine all'interno della collezione di documenti,

maggiore sarà la sua rilevanza a fronte di un'analisi di Topic Modeling. Per tale motivo possiamo adottare come metro valutativo il *weighting value*, metrica che misura proprio la frequenza di occorrenza di ogni parola in ogni singola recensione, quindi il peso che ogni termine ha all'interno della raccolta. Il passo successivo è stato creare un modello LDA a $k = 4$ topic utilizzando la funzione LDA(). Si consideri che la scelta di k è del tutto arbitraria. Per ogni recensione, l'algoritmo ha iterato il seguente processo, articolato su tre step:

1. Per ogni recensione ha selezionato un topic dalla sua distribuzione sui diversi topic.
2. Ha campionato una parola dalla distribuzione tra quelle associate al topic scelto.
3. Ha ripetuto il processo per tutte le parole nel documento.

```
lda = LDA(dtm, k = 4, control = list(seed = 1154))
```

La funzione ha restituito un oggetto contenente i dettagli completi del modello, ovvero il modo in cui le parole sono associate ai topic ed il modo con cui i topic sono associati alle recensioni. L'ultimo step è stato convertire il formato lda in un tidy data frame, nominato "topics", che ha restituito le probabilità di appartenenza *per-topic-per-word* β. Di seguito un estratto:

```
#lda beta
topics = tidy(lda, matrix = "beta")
topics
  A tibble: 17,008 x 3
    topic term        beta
    <int> <chr>      <dbl>
        1  great    1.06e-36
        2  great    3.29e-84
        3  great    1.83e-74
        4  great    9.79e- 5
        1  oh       2.12e-36
        2  oh       3.79e-45
        3  oh       8.32e- 5
        4  oh       1.03e-49
        1  thank    1.54e-10
        2  thank    7.47e-57
    ... with 16,998 more rows
```

Si noti che questa funzione ha trasformato il modello in un formato *one-topic-per-term-per-row*. Per ogni combinazione, il modello calcola la probabilità β che quel termine sia generato da quel dato topic. Per esempio, il termine "*great*" ha una probabilità di essere generato dal topic 1 pari a 1.06e-36, e una probabilità pari a 3.29e-84 di essere generato dal topic 2.

Come illustrato nella Figura 3.7, sono stati generati quattro topic utilizzando i 10 termini più comuni all'interno di ognuno di essi.

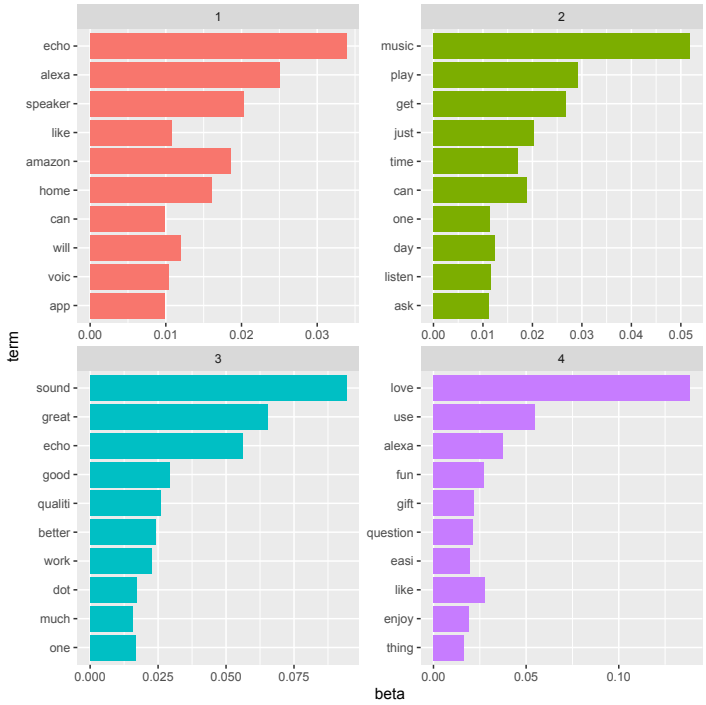

Figura 3.7. 4-topic LDA model

La visualizzazione ci permette di distinguere i quattro differenti topic estratti dalla raccolta di recensioni: le parole più comuni all'interno del topic 1 sono "*echo*", "*alexa*", "*speaker*", "*home*", "*voice*" ecc. che suggeriscono un orientamento dell'argomento discusso verso lo *speaker*, quindi verso l'utilizzo domestico del prodotto. I termini più comuni all'interno del topic 2 sono "*music*", "*play*", "*listen*" ecc. che suggeriscono un'orientamento verso la capacità del *device* di riprodurre musica (quindi verso l'ascolto). Le parole più comuni all'interno del topic 3, quali "*sound*", "*great*", "*good*", "*qualiti*", "*better*", "*work*", suggeriscono un orientamento verso la qualità del prodotto, il suono e il suo funzionamento. Il quarto ed ultimo topic, invece, raccoglie termini quali "*love*", "*use*", "*easi*", "*fun*", "*alexa*", "*enjoy*", che suggeriscono un orientamento verso la semplicità nell'utilizzo del prodotto e verso il piacere che ne deriva. Un'os-

servazione importante circa le parole in ognuno dei quattro topic è che alcune di queste possono essere comuni a più topic (vedi "alexa" presente sia nel topic 1 che nel topic 4). Questo rappresenta un vantaggio del Topic Modeling rispetto ai metodi di "*hard clustering*": i topic utilizzati nel linguaggio naturale ammettono *overlapping* di parole.

Oltre a stimare ogni topic come una combinazione di parole, l'approccio LDA modella anche ciascun documento come una combinazione di topic. Possiamo esaminare le probabilità *per-document-per-topic* γ (gamma). Di seguito un estratto:

```
#lda gamma
topics2 = tidy (lda, matrix = "gamma")
topics2
  A tibble: 14,328 x 3
    document topic gamma
    <chr>    <int> <dbl>
     1            1     0.247
     2            1     0.268
     3            1     0.787
     4            1     0.491
     5            1     0.186
     6            1     0.804
     7            1     0.656
     8            1     0.702
     9            1     0.879
    10            1     0.164
    ... with 14,318 more rows
```

Ognuno di questi valori γ rappresenta la percentuale stimata di termini contenuti in quella specifica recensione generati da quel dato topic (in questo caso dal topic 1). Per esempio, il modello stima che il 24,7% dei termini nella recensione 1 sono stati generati dal topic 1. In questo senso, scalando il discorso, è possibile immaginare che la maggior parte delle recensioni sia fondamentalmente tratta da un mix di topic. Questo però non è detto; la recensione 9, per esempio, mostra un valore γ prossimo all'1. Questo vuol dire che è tratta quasi interamente dal topic 1. Estraendo la recensione 9 dal vettore comments, quindi, mi aspetto che la stessa sia orientata verso lo *speaker* Alexa e l'impiego domestico del prodotto (argomenti del topic 1).

```
dataset$comments[9]
[1] "Overall, the Echo is an excellent product to add to your home. When
    I heard about the Echo back in 2014, I was very skeptical someone
    could come up with something useful for it to do unless you were
```

```
entrenched in the Smart Home world [...] Alexa speaker seems like a
natural part of your home.
```

Come è possibile notare, la recensione estratta copre esattamente i temi definiti dal topic 1 e mostrati nella Figura 3.7. Ora che il metodo del Topic Modeling ha generato una rappresentazione delle recensioni in un *topic space*, l'idea è utilizzare tale rappresentazione come base per performare gli algoritmi di clustering. L'obiettivo è verificare se tali algoritmi mi restituiranno un'associazione tra recensione e topic in modo similare a quanto ottenuto con il Topic Modeling.

3.2.2 Algoritmi di Clustering

Per testare e comparare i risultati ottenuti con il Topic Modeling Method, ho applicato due algoritmi di clustering: il clustering gerarchico agglomerativo e il *k*-means clustering, entrambi oggetto di analisi in questa sezione. La cluster analysis è una tecnica di analisi multivariata che consente di classificare un insieme di *pattern* (unità statistiche) in sottoinsiemi in modo che abbiano caratteristiche o proprietà simili[24]. L'obiettivo è minimizzare la "lontananza logica" interna ad ogni gruppo (devianza *within-groups*) e massimizzare quella tra i gruppi (devianza *between-groups*). Tale lontananza logica viene quantificata per mezzo di misure di similarità definite tra le unità statistiche: per dati quantitativi si utilizzano misure di distanza, dette metriche; per dati di tipo qualitativo, invece, si adottano misure *matching-type*, cioè di associazione (similarità o dissimilarità). Effettuata la scelta della misura di similarità da utilizzare, si pone la scelta dell'algoritmo di classificazione e dell'eventuale criterio di aggregazione/suddivisione.

Il problema del clustering è stato affrontato in diversi contesti e discipline e ciò riflette il suo ampio ricorso e la sua grande utilità nell'analisi esplorativa dei dati. Tale metodo di *unsupervised machine learning* si rivela come un complesso problema di combinatoria e le differenze nelle ipotesi e nei contesti relativi alle diverse comunità hanno reso lento il trasferimento di concetti e metodologie generiche utili. La varietà di tecniche per misurare la prossimità (similarità) tra gli elementi, raggrupparli e poi rappresentarli, ha prodotto un ricco e spesso confuso assortimento di metodi di clustering. Il punto di partenza è, però, capire la differenza tra analisi discriminante (classificazione supervisionata) e clustering (classificazione

[24]Jain, A.K. et al. (1999), "Data clustering: a review" in *ACM computing surveys* (CSUR), 31(3), 264-323

non supervisionata). Nel primo caso siamo provvisti di una collezione di *pattern* etichettati (pre-classificati) ed il problema consiste nell'etichettare un modello appena incontrato, ma non riconosciuto, minimizzando la probabilità di errore. In genere, i *pattern* etichettati (di *training*) vengono utilizzati per apprendere la descrizione delle classi, utilizzate poi per etichettare il nuovo modello. Nel caso del clustering, invece, il problema è raggruppare in cluster significativi una determinata raccolta di *pattern* non etichettati a due a due disgiunti in base ad un set di variabili osservate, in modo che tali cluster siano separati all'esterno (massima devianza *between-groups*) e coesi all'interno (minima devianza *within-groups*). Il clustering si rivela utile in diverse analisi esplorative del modello, quali raggruppamento, creazione di decisioni e situazioni di apprendimento automatico (estrazione di dati, recupero di documenti, segmentazione di immagini e classificazione di modelli). Tuttavia, in molti di questi problemi, sono disponibili poche informazioni preliminari sui dati e il decisore deve fare il minor numero possibile di ipotesi sugli *item*. È sotto queste restrizioni che la metodologia dell'aggregazione è particolarmente adatta all'esplorazione delle interazioni tra i dati per fare una valutazione sulla loro struttura. L'attività tipica di clustering di *pattern* include i seguenti cinque passi[25], mostrati nella Figura 3.8:

1. Rappresentazione del modello in forma di matrice dei dati, ovvero definizione del numero di classi, del numero di *pattern* disponibili e del numero, tipologia e scala delle funzioni disponibili per l'algoritmo di clustering.
2. Costruzione di una matrice di prossimità: si definisce un indice di prossimità del modello adeguato al dato principale, solitamente misurato mediante una funzione di distanza definita su coppie di modelli. La distanza euclidea ha un fascino intuitivo e viene comunemente usata per valutare la vicinanza di oggetti nello spazio bidimensionale o tridimensionale. Tale distanza funziona bene quando un set di dati ha cluster "compatti" o "isolati[26]".
3. Raggruppamento performabile secondo diversi metodi.
4. Astrazione dei dati, ovvero processo di estrazione di una rappresentazione semplice e compatta di un set di dati.
5. Valutazione dell'output, ovvero verifica ed interpretazione dei risultati.

[25]Jain, A.K. et al. (1988), "Algorithms for clustering data"

[26]Mao, J. e Jain, A. K. (1996), "A self-organizing network for hyperellipsoidal clustering (HEC)" in *Ieee transactions on neural networks*, 7(1), 16-29.

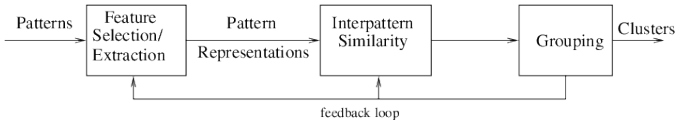

Figura 3.8. Stage di clustering

I diversi approcci ai dati di clustering possono essere descritti con l'aiuto della gerarchia mostrata nella Figura 3.9. Bisogna sottolineare che sono possibili altre rappresentazioni tassonomiche della metodologia di clustering; quella illustrata è basata sulla discussione di Jain[27].

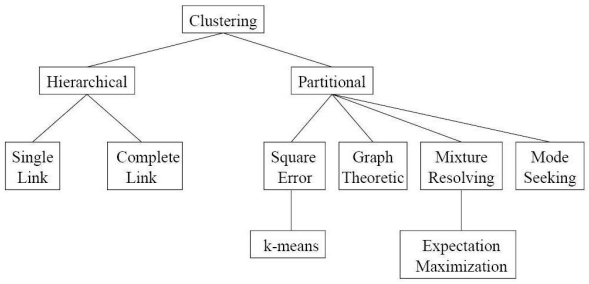

Figura 3.9. Tassonomia di clustering

Il clustering gerarchico è spesso ritratto come il miglior approccio da un punto di vista qualitativo, ma è limitato a causa della sua *quadratic time complexity*, perchè eccessivamente oneroso in termini di calcoli richiesti. Un modo per ridurre la quantità di calcoli è scegliere a priori il numero di cluster; in tal modo l'algoritmo non sarà più gerarchico, e genererà un'unica partizione delle n unità in k cluster. La tecnica delle k-medie, infatti, ha un *time complexity* lineare nel numero dei documenti, ed è pensata per produrre un numero inferiore di cluster. Alcune volte, nella letteratura, i due approcci sono stati combinati adottando un approccio ibrido tra k-means clustering e clusterng agglomerativo: il primo applicato per la sua efficienza in fase di esecuzione, il secondo per la sua qualità. In questo lavoro di tesi si è deciso di applicare entrambi gli algoritmi in modo da confrontarli e proporre una comparazione finale con il Topic Modeling Method. L'obiettivo è stato lo stesso di quello proposto con il Topic Modeling: cercare recensioni "similari" sul prodotto e raggrupparle in differenti insiemi. Basicamente, ho provato a clusterizzare le recensioni comparando la frequenza di apparizione dei termini in ogni recensione.

Per applicare i metodi di clustering, è stata utilizzata la matrice tdm costruita nella sezione 3.1 che, come visto, accoglie in input il *text corpus*. In questa matrice,

[27] Jain, A.K. et al. (1988), "Algorithms for clustering data".

ogni colonna rappresenta una recensione, ogni riga rappresenta uno dei termini presenti nelle recensioni e ogni valore della matrice contiene il numero di occorrenze di quel dato termine in quella data recensione. Nel nuovo *data frame*, nominato "freqTerms.df", ho rimosso i termini sparsi, apparsi neIle recensioni con una frequenza inferiore al 5%; in questo modo, freqTerms.df comprende solo quei termini (34) che appaiono almeno 180 volte nell'intera raccolta di recensioni.

```
freqTerms = removeSparseTerms(tdm, 0.95)
freqTerms.df = as.data.frame(as.matrix(freqTerms))
```

Il motivo di tale snellimento deriva dal fatto che, dopo il pre-processamento del *corpus*, abbiamo ottenuto complessivamente 4252 termini apparsi almeno una volta e distribuiti all'interno delle 3590 recensioni. Il problema derivante è che molti di questi termini appaiono sporadicamente nelle recensioni, quindi risultano inutili ai fini dell'analisi. Difatti, la matrice dtm calcola una *sparsity* pari al 100%; questo vuol dire che la maggior parte delle celle riporta valore pari a 0. Settando come argomento della funzione removeSparseTerms() un valore di sparse = 0.95, sono stati rimossi solo i termini per un valore di sparse maggiore di 0.95. L'argomento sparse corriponde alla soglia della frequenza relativa del documento per un termine, al di sopra del quale il termine viene rimosso. L'interpretazione esatta è, dato il generico termine i, verranno mantenuti tutti i termini per cui $df_i > N * (1 - 0.95)$, dove N è il numero di recensioni. Intuitivamente, per completezza teorica, in prossimità dell'estremo opposto (sparse = 0.1), vengono ritenuti i termini che appaiono in tutte le recensioni.

Per meglio comprendere, di seguito un campione della matrice freqTerms.df estratto casualmente:

```
        Docs
Terms   50 54 55 56 57 58 59 60 63 67
  devic  3  1  1  1  2  5  1  0  0  1
  easi   0  1  1  0  0  0  0  0  0  0
  echo   1  2  2  2  2  4  6  3  7  5
  enjoy  1  0  0  1  0  0  0  0  0  0
  fun    0  1  0  0  0  0  0  0  0  0
```

Per creare cluster di parole, impiegando sia l'algoritmo di clustering gerarchico agglomerativo che il *k*-means, sono state applicate funzioni disponibili nel pacchetto cluster().

Cluster Analysis Gerarchica

La caratteristica principale che distingue i metodi gerarchici da quelli non gerarchici è che l'assegnazione di un oggetto ad un cluster è irrevocabile. Quindi, una volta che un oggetto è entrato a far parte di un cluster, non ne viene più rimosso.

La prima partizione nei metodi di clustering gerarchici è tra metodi agglomerativi e scissori. Nel primo caso, dato un'insieme di oggetti da classificare e definita la distanza quale misura di dissimilarità, si aggregano all'interno di un cluster, in modo iterato, le unità più vicine (ovvero con distanza minima) fino alla formazione di un unico cluster contenente tutte le unità. I metodi di aggregazione impiegati possono essere il legame singolo, il legame completo, il legame medio, il metodo del centroide o, in alternativa, il metodo di Ward. Tutto il procedimento poggia sulla definizione del criterio di assegnazione delle unità ai cluster (o di un cluster piccolo ad uno più grande). Nel caso dei metodi gerarchici scissori, invece, si definiscono partizioni sempre più fini dell'insieme iniziale. L'*iter* prevede che si suddivida progressivamente l'insieme dei dati in un numero sempre crescente di sottoinsiemi, fino a quando ogni elemento distinto rappresenterà un singolo cluster.

Un algoritmo gerarchico produce un dendrogramma, ovvero un grafo ad albero rappresentante il raggruppamento annidato delle unità statistiche[28]. In particolare, il dendogramma indica i livelli di somiglianza, quindi di aggregazione delle unità: a livelli differenti, i raggruppamenti cambiano. In sostanza, l'albero prodotto visualizza l'intero processo di aggregazione, consistente in una vera e propria gerarchia di partizioni, dove ogni singola partizione si ottiene tagliando il dendrogramma ad un dato livello dell'indice di distanza della gerarchia. La scelta di quanti gruppi finali ottenere si traduce nel problema: a quale livello tagliare l'albero? Poichè l'interesse comune è avere il minor numero di gruppi possibile con massima omogeneità, si cerca di tagliare in corrispondenza dei rami più lunghi (cioè le verticali più lunghe).

Come sappiamo, l'idea di base è comparare il risultato del Topic Modeling Method con gli algoritmi di clustering; per questo motivo sono stati generati quattro cluster di termini utilizzando l'algoritmo agglomerativo. Riprendendo la matrice freqTerms.df, già mostrato precedentemente, cerchiamo di capire come ha operato l'algoritmo agglomerativo. Il nostro obiettivo è ottenere una famiglia di partizioni (precisamente 4 cluster) partendo dalla partizione banale in cui ogni termine x_i degli n a disposizione rappresenta un cluster, fino a giungere, tramite aggregazioni

[28]Steinbach, M. et al. (2000), "A comparison of document clustering techniques" in *KDD workshop on text mining* (Vol. 400, No. 1, pp. 525-526).

successive, alla partizione banale opposta, ovvero la situazione in cui tutti gli n termini sono inclusi in un unico cluster.

Partendo dalla matrice dei dati freqTerms.df, l'algoritmo deriva da questa una matrice di prossimità, ovvero un array di dimensione nxn che gode delle proprietà di non negatività, simmetria, identità e disuguaglianza triangolare. L'indice di prossimità utilizzato, in questo caso, è la distanza euclidea, calcolata tra i vettori della matrice dei dati. La distanza euclidea tra due generici punti di coordinate $(x_1; y_1)e(x_2; y_2)$ viene calcolata come radice della somma dei quadrati delle differenze tra i punti di coordinata; in formula: $D_{12} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. In seguito, l'algoritmo prevede l'unione iterata dei cluster, in base ad un determinato metodo applicato per calcolare la distanza tra i neo-cluster e le restanti unità statistiche. Nel caso in esame, il metodo applicato è stato quello di Ward, utilizzato come dato di input nella funzione hclust(). Il metodo di Ward, ad ogni passo, tende ad ottimizzare la partizione ottenuta tramite l'aggregazione di due elementi garantendo la minimizzazione della devianza *within-groups*. Ricordiamo che una partizione si considera tanto migliore quanto più le classi risultano omogenee al loro interno (minima devianza *within-groups*) e differenti l'una dall'altra (massima devianza *between-groups*). In base a ciò, l'algoritmo ricerca il salto minimo di aumento della varianza interna, ovvero ad ogni passo aggrega ad un cluster già individuato l'unità o il cluster che porta il minor incremento di varianza interna.

```
#clustering gerarchico
data.df.scale = scale(freqTerms.df)
dist = dist(data.df.scale, method = "euclidean")
hcluster = hclust(dist, method = "ward.D")
hclusterID = cutree(hcluster, k = 4))
```

Il dendogramma prodotto è mostrato nella Figura 3.10. Il *plot* ha aiutato a decidere a che livello tagliare il grafo, quindi a scegliere il numero di cluster da selezionare. Poichè l'altezza rappresenta la devianza *between-groups*, tendenzialmente il taglio ottimale avviene in prossimità del salto, ovvero dove la varianza è massima. In base a ciò la scelta migliore sarebbe tagliare l'albero in corrispondenza del primo *grouping level*. Ai fini della ricerca, però, possiamo scegliere anche il terzo *grouping level* che garantisce, comunque, grande varianza tra i cluster. Infine, utilizzando la funzione cutree() sono state raggruppate le parole in quattro cluster visivamente delimitati da *box* rossi.

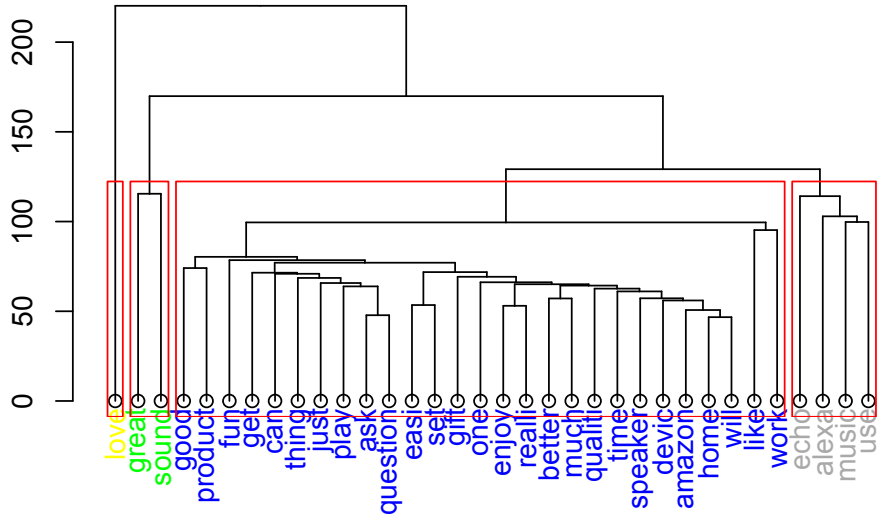

Figura 3.10. Cluster dendrogram

Al di sotto dei *box* rossi, l'altezza, che ricordiamo rappresenta la varianza tra i cluster, diminuisce in modo acuto. Per questa ragione, quattro è un buon numero di cluster da scegliere. Ogni parola è etichettata con un colore che indica l'appartenenza ad uno dei quattro cluster. Si ricordi che, come visto nella sezione3.2.1, quattro è anche il numero di distribuzioni multinominali ottenute con il metodo del Topic Modeling. Come mostrato nella Figura 3.10, del totale dei 34 termini iniziali, 27 parole appartengono al cluster blu, 2 al cluster verde, 1 al cluster giallo e 4 termini al cluster grigio.

Ora è possibile fare una comparazione tra i due differenti metodi di analisi testuale evidenziando similarità e differenze. Prima di tutto, una chiarificazione è necessaria: come sappiamo l'algoritmo di clustering è stato applicato alla matrice costituita solo da quei termini che appaiono nelle recensioni con una frequenza rilevante; nel Topic Modeling Method, invece, abbiamo usato l'intera BoW. In qualunque caso possiamo non considerare questa come una dissimilarità rilevante perchè, il Topic Modeling, utilizzando l'approccio LDA, ha creato topic in base alla frequenza di occorrenza delle parole, prendendo in considerazione solo i termini che appaiono più spesso nella raccolta di recensioni. Per quanto riguarda le somiglianze, è possibile notare come il cluster che raggruppa le parole in verde, quali "*great*" e "*sound*", corrisponde al topic orientato alla qualità del prodotto e del suono; il cluster che raggruppa le parole grigie, quali "*echo*", "*alexa*" corrisponde al topic orientato allo *speaker* e all'utilizzo domestico del *device*; il cluster che raggruppa le parole in giallo, in questo caso "*love*" corrisponde al topic orientato all'apprezzamento

del prodotto e al piacere derivante dall'utilizzo mentre il cluster con le parole in blu, quali "*listen*", "*play*" coincide con il topic orientato sulla capacità del *device* di riprodurre musica (quindi sull'ascolto).

K-means Clustering

Mentre nel caso dei metodi gerarchici l'algoritmo cerca, ad ogni passo, la migliore scissione o aggregazione tra cluster, nei metodi di clustering non gerarchico l'algoritmo partiziona le n unità in un numero prestabilito $k(<n)$ di gruppi basandosi sull'ottimizzazione di un qualche criterio predefinito[29]. L'inizializzazione dell'algoritmo avviene indicando k centri di partenza intorno ai quali aggregare le unità. A differenza dei metodi gerarchici, l'assegnazione di un oggetto ad un cluster non è irrevocabile, ovvero se l'allocazione iniziale risulta inappropriata le unità vengono riassegnate ad un diverso cluster; in questo modo, ad ogni passo, l'algoritmo rimette in discussione la partizione ottenuta. Di solito, scelta una partizione iniziale, si cerca di migliorarla in funzione del criterio di minimizzazione della varianza interna. La procedura di allocazione avviene tramite l'ottimizzazione di una funzione obiettivo espressa in termini di scomposizione della devianza totale: l'algoritmo ricerca la partizione che garantisce massima coesione interna tra le unità (minima devianza *within-groups*). Il processo prevede una prima partizione mediante l'individuazione di k poli provvisori e l'allocazione delle unità statistiche ai poli più vicini, in modo da formare i primi k cluster. Successivamente si individua computazionalmente il baricentro di ogni gruppo e, per ogni gruppo, si calcola la varianza interna considerando i baricentri appena calcolati come nuovi centri provvisori. Calcolando la distanza euclidea tra ogni unità statistica ed il suo centroide, se l'algoritmo individua che tale unità è più vicina ad un altro dei k centroidi, allora viene riassegnata. Tale procedimento di riallocazione delle unità viene iterato fino a quando l'algoritmo converge, ovvero fino a quando non ci sarà più alcun spostamento di centroide. L'algoritmo iterativo individua un punto di minimo della funzione obiettivo che definisce il numero di iterazioni necessario a minimizzare la devianza *within-groups*. Il problema è che l'algoritmo potrebbe convergere ad un ottimo locale (e non globale); ciò significa che se partissimo da un diverso insieme di centri provvisori potremmo ottenere una partizione differente. Per questo motivo, viene definita una regola di arresto, ovvero un numero massimo di iterazioni dopo il quale si fa convergere l'algoritmo. Un grosso problema del k-means è la sua forte

[29] Hartigan, J. A. e Wong, M. A. (1979), "Algorithm AS 136: A k-means clustering algorithm" in J*ournal of the Royal Statistical Society*, Series C (Applied Statistics), 28(1), 100-108.

dipendenza dal piazzamento iniziale dei centroidi che, in assenza di informazioni a priori sul set di dati, è praticamente casuale. Le prestazioni dell'algoritmo possono dunque variare di molto in base all'inizializzazione, e non si hanno garanzie sul risultato finale. In questo senso il *k*-means riscontra difficoltà all'aumentare della dimensionalità dello spazio. Diventa complicato, infatti, individuare dei gruppi di elementi simili quando i dati risultano molto dispersi nello spazio. In problemi come quelli del clustering di documenti di testo, in cui si ha una dimensionalità elevata ed in cui oltretutto oggetti di categorie diverse possono risultare simili, i punti relativi ai vari documenti sono disposti nello spazio in maniera sparsa, senza formare dei gruppi ben distinti dagli altri. Il risultato è che l'algoritmo tende a dare come soluzione un gruppo che contiene molti elementi (generalmente quello che fin dall'inizializzazione casuale ingloba più elementi e che ad ogni passo ne ingloba sempre di più) e tanti gruppi che contengono invece pochi elementi. La disposizione iniziale dei cluster favorisce insomma un gruppo che tende ad inglobare la maggior parte degli oggetti penalizzando di conseguenza tutti gli altri. Per arginare questo problema abbiamo ridotto la dimensionalità dello spazio dei dati adottando come argomento la matrice contenente i termini più frequenti (freqTerms.df). L'obiettivo è stato eliminare le componenti poco significative in modo da poter definire la similarià semantica tra i documenti e rendere più semplice la separazione degli elementi tra loro "simili".

Per performare l'algoritmo delle *k*-medie sulla matrice dei dati freqTerms.df, anzitutto, è stato specificato il numero *k* di centroidi iniziali, dove *k* è determinato a priori. Si consideri che, applicando l'algoritmo *k*-means, non è possibile raggruppare i documenti in modo efficace se il criterio di suddivisione non è noto a priori. Nel caso in esame, poichè l'obiettivo è operare un confronto con i risultati ottenuti con il clustering gerarchico agglomerativo e, in seconda battuta, con quanto ottenuto con il Topic Modeling Method, ho fissato $k = 4$, prevedendo così un partizionamento finale dei dati in quattro *subset*. Uno sguardo più da vicino, potrebbe aiutare nella comprensione dell'algoritmo. Di seguito un estratto della matrice freqTerms.df:

```
1          Docs
2 Terms    50  54  55  56  57  58  59  60  63  67
3   devic   3   1   1   1   2   5   1   0   0   1
4   easi    0   1   1   0   0   0   0   0   0   0
5   echo    1   2   2   2   2   4   6   3   7   5
6   enjoy   1   0   0   1   0   0   0   0   0   0
7   fun     0   1   0   0   0   0   0   0   0   0
```

In primo luogo, l'algoritmo prevede la scelta casuale di quattro centroidi (cluster iniziali). Supponiamo siano stati selezionati i documenti 54, 55 e 57. Lo step successivo è calcolare la distanza euclidea tra i cluster iniziali e tutti gli altri documenti contenuti nella raccolta. La funzione applicata, kmeans(), accoglie come argomento la matrice delle distanze "dist", già costruita per il clustering gerarchico nella sezione precedente.

```
#k-means clustering
kmeans4 = kmeans(dist, centers = 4, nstart = 100)
```

Seguendo il processo iterativo di riallocazione, l'algoritmo genera quattro cluster, ognuno dei quali, al proprio interno, contiene un certo numero di osservazioni (termini). Come mostrato nella Figura 3.11, del totale dei 34 termini iniziali, 25 parole appartengono al cluster blu, 2 al cluster verde, 1 al cluster viola e 6 termini al cluster rosso.

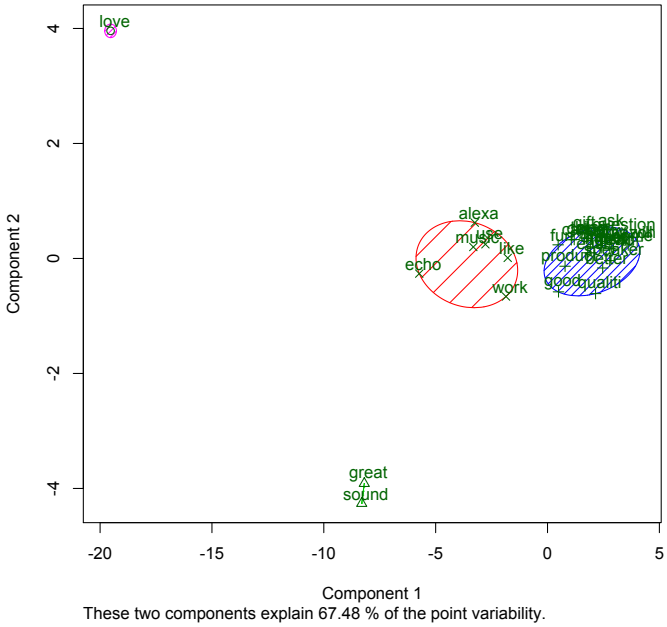

Figura 3.11. Clusplot dell'algoritmo k-means

Osservando la Figura 3.11 è possibile definire le corrispondenze e le dissimilarità rispetto al modello Topic Modeling. La prima dissimilarità è che, per applicare l'algoritmo delle *k*-medie, è stata utilizzata una matrice costituita solo dai termini non sparsi, mentre nel Topic Modeling è stata impiegata l'intera BoW. In realtà, come già spiegato nella sezione precedente, si tratta di una falsa dissimilarità.

Le corrispondenze con il metodo del Topic Modeling sono risultate le stesse del clustering gerarchico: il cluster verde, che raggruppa le parole "*great*" e "*sound*", corrisponde al topic orientato verso la qualità del prodotto e del suono; il cluster rosso che raggruppa parole quali "*echo*", "*alexa*" corrisponde al topic orientato verso lo *speaker* e l'utilizzo domestico del prodotto; il cluster rosa, cui corrisponde la parola "*love*", coincide con il topic orientato verso l'apprezzamento del prodotto e il piacere derivante dall'utilizzo; il cluster blu che clusterizza parole quali "*listen*" e "*play*", infine, coincide con il topic orientato verso la capacità del *device* di riprodurre musica (quindi l'ascolto).

3.3 Sentiment Analysis

Oggigiorno, milioni di utenti contribuiscono quotidianamente all'accumulo di dati su Internet e, tale ricchezza di contenuti, diventa sempre più importante fonte di informazione per la raccolta ed il monitoraggio delle opinioni dei consumatori. Come visto nella sezione 1.2, la *sentiment analysis* è quella sottodisciplina della linguistica computazionale, dal recente successo, che si focalizza sull'opinione espressa in un documento testuale. La gestione ed interpretazione di tali contenuti fornisce importanti implicazioni per le aziende: analizzando e classificando le opinioni, è possibile monitorare l'atteggiamento mutevole del pubblico. La questione essenziale è identificare come i sentimenti sono espressi nei documenti testuali e se le espressioni indicano opinioni positive o negative nei confronti del prodotto. Pertanto, l'analisi del *sentiment* implica l'identificazione di tre elementi tra loro correlati: espressioni di *sentiment*, polarità e forza delle espressioni, relazione tra *sentiment* e prodotto. La maggior parte del lavoro relativo all'analisi del *sentiment* fino ad oggi si è concentrato sull'identificazione delle espressioni di *sentiment* e della loro polarità. Più nel dettaglio, è possibile identificare diverse sottoattività, tutte relative al *tagging* di un determinato documento in base all'opinione espressa[30]:

1. Determinare la soggettività del documento, ovvero decidere se un dato testo ha una natura fattuale (cioè descrive una data situazione o evento, senza esprimere un'opinione positiva o negativa su di esso) o esprime un'opinione sull'argomento. Ciò equivale ad eseguire una categorizzazione binaria del testo in due classi: obiettivo e ooggettivo.

[30]Pang, B. e Lee, L. (2004), "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts" in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics* (p. 271), Association for Computational Linguistics.

2. Determinare l'orientamento (o la polarità) del documento, ovvero decidere se un dato testo esprime un'opinione positiva o negativa sull'argomento.
3. Determinare la forza dell'orientamento del documento, ovvero definire il grado di intensità dell'opinione. Ad esempio, nel caso di un'opinione positiva, determinare se la stessa è categorizzabile come leggermente positiva, positiva o fortemente positiva.

In questo lavoro di tesi, le applicazioni computazionali verteranno sui punti 2 e 3, ovvero sul determinare la polarità generale delle recensioni rispetto al prodotto in esame quantificandone l'intensità. Il primo passo, illustrato nella sezione 3.3.1 è stato estrarre, tramite gli strumenti offerti dal pacchetto tidytext tutte le *sentword* contenute nel corpo delle recensioni in modo da disporre di una prima misura indicativa sulla polarità della raccolta. Nella sezione 3.3.2 invece, l'obiettivo è stato identificare, quantificare e caratterizzare il contenuto sentimentale dell'unità di testo, fornendo un'interpretazione realistica del mercato. Il confronto tra i risultati ottenuti con tre differenti lessici ci ha fornito una prospettiva a più ampio raggio per meglio comprendere l'orientamento emozionale delle recensioni.

Questa analisi ci consentirà di ottenere informazioni generate direttamente dai recensori, fornendo *insight* quantificati sull'*impression* generale che i consumatori hanno circa il prodotto.

3.3.1 Sentiment Words

Come di seguito spiegato, il primo step è stato ottenere una lista delle *sentiment word* presenti nellle recensioni con relative frequenze di occorrenza, in modo da avere un'idea generale sulla polarità della raccolta senza aver ancora assegnato alcun punteggio. Questo è stato possibile utilizzando strumenti dei pacchetti tidytext e dplyr.

Prima di tutto, è necessario filtrare la lista delle parole a partire dalle righe del vettore comments (ovvero dalle recensioni.) L'obiettivo, in pratica, è ricavare una matrice in cui ad ogni riga corrisponde una parola. Tale tabella di contingenza, in realtà, l'abbiamo già creata convertendo il *corpus* in una *term-document matrix* nella sezione 3.1. Poichè gli oggetti della matrice tdm non possono essere utilizzati direttamente con gli strumenti del pacchetto tidytext, bisogna convertire la matrice tdm in un *data frame* di classe tidy, ovvero *one-token-per-document-per-row*, utlizzando l'apposita funzione di conversione tidy(). Un *token* è un'unità di testo significativa che ci interessa utilizzare per l'analisi e la *tokenizzazione* rappresenta il processo di divisione del testo in *token*. Per l'estrazione del testo, il *token* memorizzato in

ogni riga è spesso una singola parola, ma può anche essere un *n-gram*, una frase o un paragrafo. Il *data frame* ottenuto è stato nominato "tidy". Tale oggetto ha restituito un *tibble*, ovvero una classe di *data frame* che non converte le stringhe in fattori e non utilizza nomi di riga. La dimensione del *tibble* è 43475x3 variabili dove, come mostrato nell'estratto sotto, ognuna delle 43475 righe corrisponde ad uno specifico termine mentre le 3 variabili sono il termine, la recensione di appartenza ed il conteggio di occorrrenza.

```
> tidy
# A tibble: 43,475 x 3
   term      document count
   <chr>     <chr>    <dbl>
 1 abil      1            1.
 2 accept    1            1.
 3 add       1            1.
 4 alex      1            1.
 5 alexa     1            1.
 6 allot     1            1.
 7 allow     1            1.
 8 also      1            2.
 9 although  1            2.
10 amazon    1            1.
# ... with 43,465 more rows
```

Dopo questo step di conversione, è possible individuare le *sentiment word* e categorizzarle come positive o negative utilizzando, come dato di input nella funzione inner_join(), il lessico bing. Il lessico bing, come meglio vedremo nella sezione 3.3.2, consta di 6788 termini e categorizza le parole in positive e negative secondo una classificazione binaria: -1 se negativa, +1 se positiva.

```
#sentiment words
sentwords = tidy%>%
inner_join(get_sentiments("bing"), by = c(term = "word"))
```

Il *data frame* sentwords è una matrice di classe *tibble* con dimensioni 6888x4, dove 6888 rappresenta il numero di righe, ovvero il numero di termini di *sentiment* che figurano nelle recensioni mentre 4 è il numero di variabili: il termine, il conteggio di occorrenza, la recensione di appartenenza del dato termine ed il *sentiment*, che indica il modo in cui ogni parola è stata etichettata stando alla classificazione binaria

positivo/negativo. In particolare, delle 6888 *sentiment word* che figurano nell'intero corpo di recensioni, 5827 sono state etichettate come positive, 1061 come negative. Questo già ci dà un'idea sull'orientamento delle recensioni rappresentando una misura indicativa della polarità della raccolta.

```
> sentwords
# A tibble: 6,888 x 4
   term      document count sentiment
   <chr>     <chr>    <dbl> <chr>
 1 better    1            2. positive
 2 concern   1            1. negative
 3 enjoy     1            1. positive
 4 free      1            1. positive
 5 hack      1            1. negative
 6 lack      1            1. negative
 7 like      1            2. positive
 8 nice      1            2. positive
 9 restrict  1            1. negative
10 richer    1            1. positive
# ... with 6,878 more rows
```

La Figura 3.12 ci permette di visualizzare quali parole delle recensioni contribuiscono maggiormente al *sentiment* positivo o negativo. Questo impatto è misurato semplicemente considerando la frequenza di occorrenza del termine (valore in ordinata della Fig. 3.12): più si ripete all'interno della raccolta delle 3590 recensioni, maggiore sarà il suo contributo al *sentiment* generale.

Le parole positive più comuni mostrate dalla Figura 3.12 sono "*love*"(che appare più di 1500 volte), "*great*"(più di 750 volte) e "*like*"(più di 500 volte), mentre i termini negativi che occorrono con maggior frequenza sono "*problem*"(poco più di 100 volte) e "*disappoint*"(poco più di 100 volte). Considerando gli estremi a titolo esemplificativo, possiamo notare come le parole di *sentiment* positivo, oltre ad essere in numero maggiore, appaiono con frequenza maggiore rispetto a quelle di *sentiment* negativo suggerendoci l'andamento nella polarità della raccolta. Per ognuna di queste *sentiment word*, inoltre, è possibile assegnare un punteggio in modo da determinare la polarità di ogni recensione e, conseguentemente, dell'intero corpo di recensioni. Questo processo è oggetto di discussione nella prossima sezione 3.3.2.

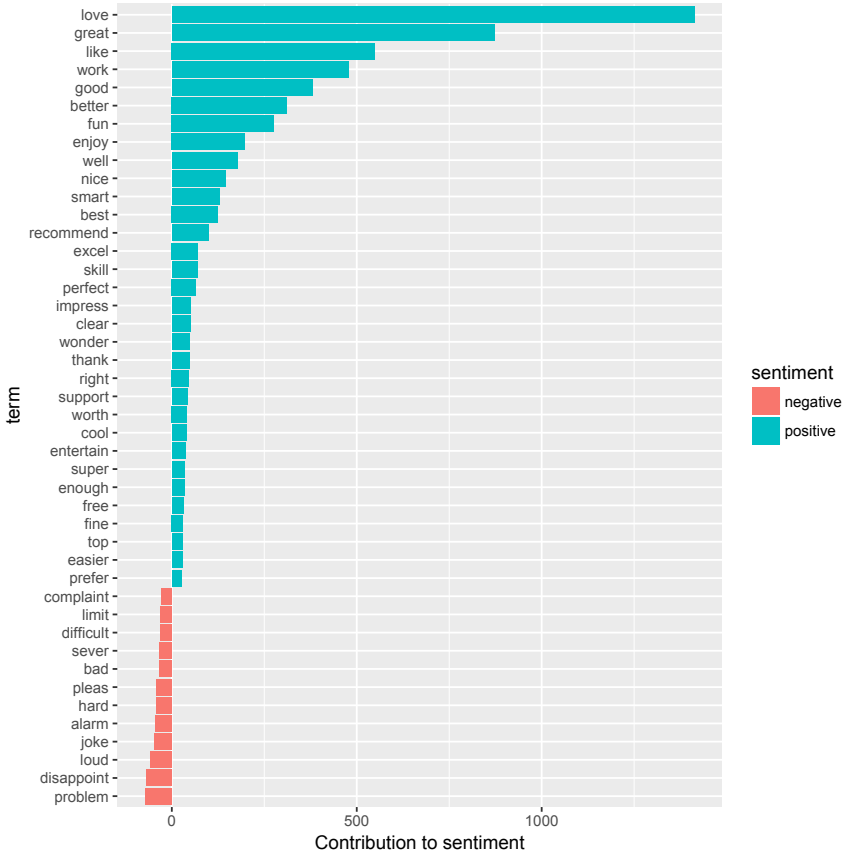

Figura 3.12. Termini con la maggior contribuzione al sentiment

3.3.2 Sentiment Polarity

In questa sezione vediamo come assegnare un punteggio ad ogni recensione, fornendo una misura più dettagliata della polarità rispetto al mero rapporto tra *sentiment word* positive e negative e relativa frequenza di occorrenza, oggetto di trattazione nella precedente sezione 3.3.1. Per eseguire l'analisi è possibile applicare funzioni del pacchetto syuzhet che estraggono il *sentiment* dal testo tramite l'impiego di differenti dizionari basati su unigrammi, quindi singole parole[31].

Per assegnare punteggi alle parole di *sentiment* positivo/negativo, il pacchetto syuzhet offre tre differenti lessici: "afinn", "bing" e "nrc", illustrati nella Tabella 3.3. Il primo consta di 2476 termini ed assegna alle parole un punteggio tra -5 (molto negativo) e 5 (molto positivo); il secondo conta 6788 termini e categorizza i termini in positivi o negativi secondo una classificazione binaria (-1 se negativo, +1 se positivo). Il lessico nrc, invece, conta al suo interno 13901 parole e le categorizza

[31] Özdemir, C. e Bergler, S. (2015), "Clac-sentipipe: Semeval2015 subtasks 10 b, e, and task 11" in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 479-485).

in dieci classi emotive: positivo, negativo, rabbia, eccitazione, disgusto, paura, gioia, tristezza, stupore e fiducia. Da dove derivano questi tre lessici di *sentiment*? Sono stati costruiti tramite *crowdsourcing* con il lavoro di più autori, e sono stati convalidati utilizzando nuovamente alcune combinazioni di *crowdsourcing*. E' bene ricordare che non tutte le parole inglesi sono inserite all'interno dei tre lessici perché molte di queste sono neutre. È altrettanto importante tener presente che questi metodi non tengono conto dei qualificatori prima di una parola (come nel caso "non buono" o "non vero") ma si basano esclusivamente sugli unigrammi. Inoltre, non è possibile identificare in modo automatico nelle recensioni sentimenti come sarcasmo ed ironia. Un'ultima nota riguarda la dimensione del frammento di testo su cui sommiamo i punteggi assegnati agli unigrammi e il suo possibile effetto sull'analisi. A differenza dei testi corti, ridotti ad una frase o a un paragrafo, i testi che constano di molti paragrafi possono avere come risutato un punteggio mediamente intorno allo 0, magari non veritiero. Considerando tutte le limitazioni del tipo di analisi, l'*overload* di termini di *sentiment* può portare ad un'errata classificazione generale.

L'analisi del *sentiment* delle recensioni (quindi della polarità) richiede, da un punto di vista logico, di considerare il testo come una combinazione delle proprie parole ed il contenuto sentimentale delle 3590 recensioni come la somma del contenuto sentimentale dei termini singoli. Da un punto di vista computazionale, inserendo uno dei tre lessici disponibili nel pacchetto syuzhet come dato di input alla funzione get_sentiment(), ad ogni termine presente nelle recensioni del vettore comments viene assegnato un punteggio secondo i criteri dello specifico lessico.

```
1 afinn = get_sentiment(dataset$comments,method="afinn")
2 bing = get_sentiment(dataset$comments,method="bing")
3 nrc = get_nrc_sentiment(dataset$comments)
```

Cumulando secondo somma algebrica i punteggi assegnati alle parole che compongono ogni singola recensione, abbiamo ottenuto, in corrispondenza di questa, un valore totale di *sentiment*, in grado quindi di quantificarne la polarità. La Tabella 3.3 mostra, per ogni lessico, la funzione del pacchetto syuzhet applicata, il punteggio totale corrispondente e altre descrittive come media, mediana, valore minimo, valore massimo e deviazione standard.
In corrispondenza delle celle del lessico nrc non ci sono punteggi; in effetti non è possibile calcolare un punteggio totale delle recensioni utilizzando tale metodo. Il motivo è che questo lessico, per sua natura, non assegna punteggi alle parole di *sentiment* ma semplicemente le categorizza associandole ad una delle 10 classi

Lessico	Funzione	Punteggio	Min	Mediana	Media	Max	SD
AFINN	get_sentiment itera sul vettore comments e restituisce i valori di sentiment in base al metodo "afinn"	15044	-12	3	4.2	6	3.9
BING	get_sentiment itera sul vettore comments e restituisce i valori del sentiment in base al metodo "bing"	5999	-7	1	1.67	14	1.69
NRC	get_nrc_sentiment calcola la presenza di otto differenti emozioni e la rispettiva valenza all'interno del vettore comments	-	-	-	-	-	-

Tabella 3.3. Descrizione dei lessici e risultati di polarità

emotive precendentemente viste. Il *barplot* in Figura 3.13 mostra la distribuzione delle otto emozioni fornite dal lessico nrc all'interno del *corpus* di recensioni. In particolare, l'asse delle ascisse indica la percentuale delle recensioni caratterizzate dalla specifica classe emotiva annoverata in ordinata.
Il risultato è che sentimenti positivi quali gioia, eccitazione e gioia prevalgono di

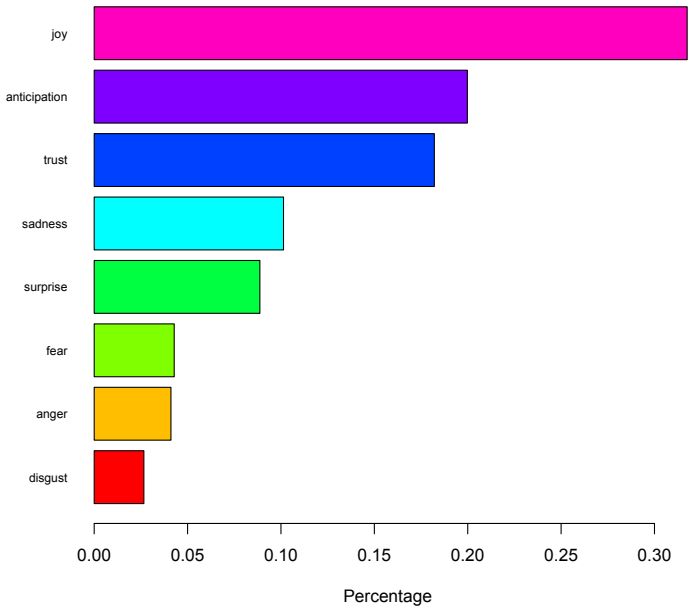

Figura 3.13. Barplot della distribuzione delle emozioni

gran lunga sui sentimenti negativi. Anche questa rappresenta una misura della polarità delle recensioni, espressa in forma diversa rispetto all'output fornit dai metodi afinn e bing, ma altrettanto indicativa. In questo senso, per quanto i tre diversi lessici forniscano risultati differenti in senso assoluto, mostrano comunque traiettorie relative simili.

Per fornire una raffigurazione della *ratio* tra recensioni positive e recensioni negative, ogni documento è stato cateogrizzato secondo classificazione binaria positiva/negativa in base al lessino afinn. In particolare, se il punteggio afinn assegnato alla recensione è risultato maggiore di 0, allora la recensione apparterrà alla classe positiva; se invece il punteggio corrispondente è risultato inferiore di 0, allora la recensione apparterrà alla classe opposta. Il motivo per cui è stato utilizzato il criterio di assegnazione punti del metodo afinn per tale categorizzazione binaria invece del criterio bing deriva dal fatto che, nel primo caso, l'assegnazione è ponderata rispetto al termine. Si consideri il caso in cui in una frase appaia un termine estremamente positivo con punteggio afinn pari +5 (ad esempio "*amazing*") e due termini leggermente negativi con punteggio afinn pari a -1 cadauno. In teoria, la recensione dovrebbe risultare complessivamente positiva. Applicando, difatti, il lessico afinn, la somma algebrica pari a +3 riflette la reale polarità positiva della recensione. Se invece applicassimo il metodo bing che, come mostrato nella Tabella 3.3, assegna punteggio +1 al termine positivo e -1 a quello negativo, la somma

algebrica (pari a -1) categorizzerebbe la recensione come negativa, quando in realtà non lo è. La *ratio* tra recensioni etichettate come positive e negative è mostrata nella Figura 3.14.

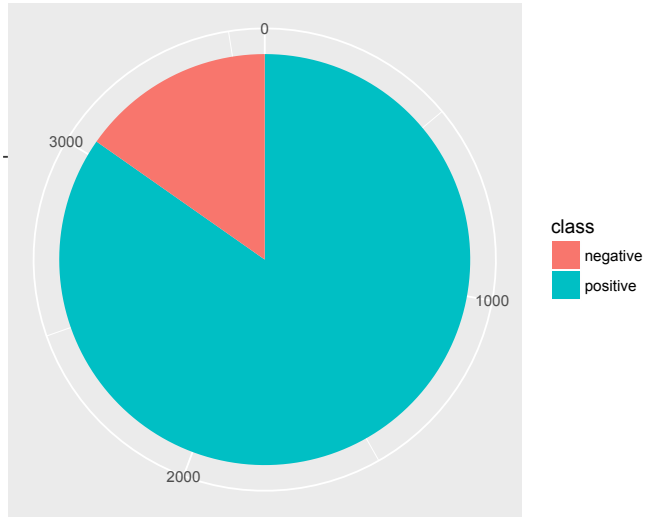

Figura 3.14. Pie chart tra recensioni positive e negative

Del totale di 3590 recensioni, 3042 risultano positive (85%) mentre 548 risultano negative (15%). Questa misura è di certo indicativa di una spiccata pendenza verso un *overall sentiment* estremamente positivo nei confronti del prodotto.

3.4 Algoritmi di Supervised Machine Learning

Nell'ambito del *text mining*, risulta di particolare efficacia applicare tecniche di *supervised machine learning* volte ad inferire una funzione o addestrare un classificatore su *training data*. Come discusso nella sezione 1.2 dedicata all'*opinion mining*, le aziende sono sempre più preoccupate per la propria reputazione online dovendo al contempo gestire una mole immensa di dati non strutturati. Riuscire a monitorare e distillare le informazioni contenute in tale archivio testuale risulta praticamente impossibile. In quest'ottica, gli algoritmi di classificazione possono rilevarsi strumenti efficienti per automatizzare il processo di estrazione dei contenuti ed operare un *tracking* in tempo reale. Costruendo classificatori di testo, l'azienda può formulare predizioni su dati mai visti e, come nel caso specifico, su nuove recensioni del prodotto. Ma cosa è un classificatore? Un *classifier* è una funzione che mappa

un'istanza non etichettata in un'etichetta usando strutture di dati interni[32].

In questa sezione l'obiettivo è costruire modelli (utilizzando metodi di regressione logistica, CART e Random Forest) in grado di classificare correttamente una nuova recensione scritta per il prodotto (assegnandole un'etichetta di appartenenza: positiva o negativa) e vautare, per ognuno di questi classificatori, l'accuratezza e le capacità di generalizzazione. Tutto ciò tenendo a mente che l'accuratezza del modello coincide con la probabilità di classificare correttamente un'istanza scelta randomicamente (dove la distribuzione di probabilità sullo spazio dell'istanza è la stessa di quella utilizzata per selezionare le istanze per il *training set* dell'*inducer*). In più verrà offerto un confronto tra le prestazioni dei vari *classifier* e tra i singoli modelli e la *baseline*. La variabile categorica utilizzata come variabile dipendente nei modelli è stata ottenuta per mezzo della classificazione della polarità a livello di recensione affrontata nella sezione 3.3 in cui, come visto, l'obiettivo è stato identificare, quantificare e caratterizzare il contenuto sentimentale dell'unità di testo. Infine, utilizzando la *k-fold cross validation* testaremo il modello CART nella fase di addestramento in modo da verificarne il sovradattamento e per fornire un'idea di come tale modello sarà generalizzabile a dati indipendenti.

3.4.1 Modello di Regressione Logistica

In questa sezione vedremo come costruire un modello di regressione logistica risolvendo il problema di classificazione preannunciato. ll modello di regressione logistica viene utilizzato quando si è interessati a studiare o analizzare la relazione causale tra una variabile dipendente dicotomica (codificata come 0-1) e una o più variabili indipendenti (predittori) [33]. Gli obiettivi possono essere molteplici:

- Individuare tra le variabili indipendenti quelle a maggior potere esplicativo, da interpretare quindi come determinanti dell'appartenenza ad una classe o all'altra.

- Ricercare la combinazione lineare delle variabili indipendenti che meglio discrimina tra il gruppo delle unità appartenenti alla classe 0 e quello delle unità appartenenti alla classe 1.

[32]Kohavi, R. (1995), "A study of cross-validation and bootstrap for accuracy estimation and model selection" in *Ijcai* (Vol. 14, No. 2, pp. 1137-1145).

[33]Harrell, F. E. (2001), "Ordinal logistic regression" in *Regression modeling strategies* (pp. 331-343). Springer, New York, NY.

- Stimare la probabilità di appartenenza ad una delle due classi per una nuova unità statistica su cui è stato osservato il vettore di variabili e, fissato per tale probabilità un valore soglia, classificare l'unità.

Quando la variabile dipendente y è dicotomica, da un punto di vista matematico è più appropriato applicare un modello non lineare; questo perchè la distribuzione teorica di riferimento non è una Normale (come accade quando la y è continua) ma una Bernoulliana. La spiegazione sta nel fatto che, nella sua formulazione, il modello lineare implica che il valore che la variabile dipendente può assumere non è limitato nè superiormente nè inferiormente; ma se la variabile dipendente è dicotomica, e può assumere solo valori all'interno dell'intervallo [0; 1], la curva che rappresenta la relazione tra x e y deve essere di tipo logistico. Ciò che interessa, infatti, non è il valore predetto (come nella regressione lineare), bensì la probabilità che un dato elemento appartenga ad una delle due classi. Un modo per risolvere il dilemma dell'assegnazione dei valori ai livelli è quello di sostituire la probabilità (ad esempio di y = 1) con l'*odds*. L'*odds* è un modo di esprimere la probabilità mediante un rapporto tra le frequenze osservate in una classe e le frequenze osservate nell'altra, ovvero $\frac{p}{1-p}$. Per semplice trasformazione, l'equazione di regressione logistica può essere scritta in termini di un rapporto di probabilità:

$$\frac{p}{1-p} = \exp(b_0 + b_1x_1 + b_2x_2 + ... + b_px_p)$$

Il coefficiente b_0 rappresenta l'intercetta della curva sull'asse delle ordinate, mentre il coefficiente b_i definisce la pendenza della curva. Prendendo il logaritmo naturale del rapporto tra le rispettive probabilità condizionate di appartenere alle due classi, possiamo riscrivere l'equazione in termini di *logit*. Il *logit* rappresenta una funzione lineare dei predittori.

$$\ln(\frac{p}{1-p}) = (b_0 + b_1x_1 + b_2x_2 + ... + b_px_p)$$

Il coefficiente b_i indica quanto il *logit* cambia al variare unitario della x_i. Manipolando la formula precendente, si ottiene:

$$p = \frac{1}{1+e^{-(b_0+b_1x_1+b_2x_2+...+b_px_p)}}$$

Utilizzando il *logit*, si costruisce la curva logistica mostrata nella Figura 3.15. Il grafico [34] della funzione descrive una curva monotona a forma di S allungata (detta "sigmoide"), limitata superiormente dalla retta y=1 ed inferiormente dalla retta y=0, alle quali tende asintoticamente.

[34]Sayad S. (2017), "Logistice Regression". Accessibile da: *http://www.saedsayad.com/logistic_regression.htm*.

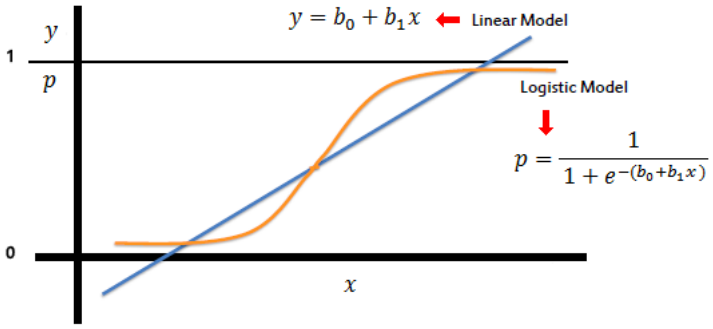

Figura 3.15. Curva logistica

Come accennato nell'introduzione della sezione, l'obiettivo è costruire un *classifier*, valutarne l'accuratezza ed infine confrontarlo con la *baseline* (modello che fornisce come previsione sempre il valore più frequente) e con altri *classifier*. Per costruire tale modello sul nostro set di dati è necessario, anzitutto, definire la variabile dipendente dicotomica ed il set di predittori.

Dalla *document-term matrix*, nominata dtm nella sezione 3.2.1, sono stati rimossi i termini sparsi, rinominando la matrice "frq.words". In questa matrice ogni riga rappresenta una recensione, ogni colonna rappresenta un termine ed ogni valore della matrice (cella) contiene il numero di apparizioni di quel dato termine all'interno di quella data recensione. Le parole che compongono la matrice sono quelle che appaiono nelle recensioni almeno il 2% delle volte. In questo modo, frq.words è una matrice di dimensione 112x3590 che consiste solo di quei termini (112) che appaiono almeno 72 volte nell'intera raccolta di recensioni. Il motivo di tale snellimento deriva dal fatto che, dopo il pre-processamento del *corpus*, abbiamo ottenuto complessivamente 4252 termini distribuiti all'interno delle 3590 recensioni. Il problema derivante è che molti di questi termini appaiono sporadicamente nelle recensioni, quindi risultano inutili ai fini dell'analisi.

```
1 sparse = removeSparseTerms(dtm, 0.98)
2 frq.words = as.data.frame(as.matrix(sparse))
```

La logica è la stessa seguita nella sezione 3.2.2 dove, per applicare gli algoritmi di clustering, abbiamo utilizzato la matrice tdm composta solo dai termini più frequenti. Per il modello di regressione logistica, però, utilizzeremo la dtm, matrice trasposta della tdm. Per ottenere come variabile dipendente una variabile binaria che assuma valore pari a 1 quando la recensione è positiva e valore 0 nel caso

complementare, anzitutto il dataset è stato combinato con il *data frame* afinn che, come visto nella sezione 3.3.2, indica per ogni recensione la somma algebrica dei punteggi assegnati ad ogni *sentiment word* all'interno dell'intervallo [-5; +5]. Il motivo per cui per tale classificazione è stato utilizzato il metodo afinn, come spiegato nella sezione 3.3.2 si basa, essenzialmente, sul fatto che il lessico afinn assegna i punteggi pesando i termini. Così facendo, si evitano errate classificazioni che potrebbe essere indotte dall'utilizzo del metodo bing. Il passo successivo è stato creare il vettore "Positive" che assume valore TRUE quando l'afinn *score* di una recensione è maggiore di 0, valore FALSE nel caso complementare. Per ottenere un vettore di valori 1 e 0, l'ultimo step è stato sostituire TRUE con il valore 1 (che definisce l'appartenenza alla classe positiva) e FALSE con il valore 0 (che definisce l'appartenenza alla classe negativa). Questo vettore, nominato "positive", è stato inserito nella matrice frq.words, utilizzata per allenare il classificatore e poi verificarne la generalizzazione. La Tabella 3.4 mostra la composizione aggiornata del dataset.

	classe	**content**
comments	character	Testo della recensione
stars	numeric	Punteggio assegnato al prodotto dal recensore su una rating scale a 5 punti
nchar	integer	Numero di caratteri (come intero positivo) usato da ciascun recensore
afinn	integer	Punteggio come intero positivo assegnato ad ogni recensione in base al lessico afinn
Positive	character	Vettore che assume valore TRUE se afinn>0, FALSE se afinn<0
positive	integer	Vettore che assume valore 1 se la recensione è positiva, 0 se negativa

Tabella 3.4. Categorie informative nel nuovo dataset

La separazione dei dati in *training set* e *testing set* (o *validation set*) rappresenta una parte importante della valutazione dei modelli di *data mining*. Il *training set*, costituito dalla maggior parte dei dati viene utilizzato per elaborare il modello poi viene testato eseguendo stime sul *testing set*[35].

[35]Shao, Y. e Lunetta, R. S. (2012), "Comparison of support vector machine, neural network, and

"Idealmente, il modello dovrebbe essere valutato su campioni che non sono stati utilizzati per costruire o perfezionare il modello, in modo che forniscano un senso imparziale dell'efficacia del modello. Quando si dispone di una grande quantità di dati, possiamo mettere da parte una serie di campioni per valutare il modello finale. Il set di dati addestramento è il termine generale per i campioni utilizzati per creare il modello, mentre il set di dati test o convalida viene utilizzato per qualificare le prestazioni[36]".

La logica di base è che la valutazione dell'abilità del modello sul set di dati di addestramento risulta parziale. Pertanto, il modello deve essere valutato su un campione di dati simili in modo da fornire una stima imparziale dell'abilità del classificatore. Utilizzando, per il *validation set*, dati simili a quelli che compongono il *training set*, è possibile ridurre al minimo gli effetti delle discrepanze di dati e comprendere meglio le caratteristiche del modello. L'esame di validità sul *testing set* ci fornirà informazioni sulle capacità del modello di generalizzare ad *out-of-sample data*.

Per allenare il classificatore probabilistico e poi validarlo, il vettore Positive è stato diviso in due *subset* randomicamente basati sulla variabile risposta utilizzando la funzione sample.split() del pacchetto caTools. Tale funzione accetta due argomenti in input: un vettore (in questo caso il vettore positive della matrice frq.words) ed una *ratio* per la divisione, settata a 7:3. Il primo *subset*, costituito dal 70% delle osservazioni (2513 recensioni) rappresenta il *training set*, ovvero la dimensione di allenamento; il secondo *subset*, costituito dal restante 30% delle osservazioni (1077 recensioni), invece, rappresenta il *testing set*, dimensione utilizzata per verificare la correttezza del classificatore. E' bene ricordare che con lo *splitting*, la funzione sample.split() preserva le relazioni originarie dei differenti valori del vettore; in questo modo, poiché nei dati di *testing* sono contenuti già valori noti per l'attributo di cui si desidera eseguire la stima nel rispetto delle proporzioni originarie, la correttezza delle ipotesi del modello può essere determinata facilmente. Dopo aver costruito il modello sul set di dati di allenamento, lo stesso è stato testato sul *validation set* eseguendovi stime in modo da valutare le capacità discriminative del classificatore costruito. Per generare il modello di regressione logistica, nominato logReg, è stata utilizzata la funzione glm(). La funzione accoglie in input la variabile dipedente (vettore positive), il set di variabili indipendenti (tutte i termini di frq.words), il dataset in cui i dati sono contenuti (*training set*) e la tipologia (binomiale). In

CART algorithms for the land-cover classification using limited training data points" in *ISPRS Journal of Photogrammetry and Remote Sensing*, 70, 78-87.

[36]Kuhn, M. e Johnson, K. (2013), "Applied predictive modeling" (Vol. 26). New York: Springer.

particolare, è la frequenza di occorrenza dei termini corrispondenti alle recensioni etichettate come positive o negative (celle della matrice) a determinare l'influenza che le singole variabili indipendenti hanno sulla variabile risposta, quindi sul determinare se la recensione sarà positiva o meno.

```
#regressione logistica
logReg = glm(positive ~ ., data=train, family=binomial)
```

Costruito il modello, è possibile analizzare il *fitting* ed interpretare il risultato. L'*ouput*, visionabile nell'Appendice A, mostra che, ad un intervallo di confidenza del 95%, alcune variabili indipendenti risultano statisticamente significative con un *p-value* $\Pr(>|z|) < .05$, altre invece no. Tra le prime figurano 40 termini su un totale di 112 come, per esempio: "*enjoy*" con Pr(>|z|)=4.50e-06, "*easi*", con Pr(>|z|)=3.75e-07, "*great*" con Pr(>|z|)< 2e-16, "*love*" con Pr(>|z|)< 2e-16 che, intuitivamente, suggeriscono una forte associazione con la probabilità che la recensione sia positiva. Come sappiamo, però, per un problema di classificazione come questo non ha senso parlare di coefficienti; è necessario, invece, generare un vettore di probabilità il cui risultato sia un numero compreso tra 0 e 1. Operando la funzione predict() sul *testing set*, la predizione ha restituito un vettore di probabilità di appartenenza alla classe 1, con un elemento per ogni misurazione. In pratica, ad ognuna delle 1077 recensioni che compongono il *testing set* è associata la probabilità che la stessa sia positiva. Utilizzando la funzione tapply() si è ottenuta la probabilità media stimata per un gruppo di misurazioni etichettate come 1 (positive) e per il gruppo di misurazioni etichettate come 0 (non positive).

```
#prediction on testing set
predictTest = predict(logReg,type="response",newdata=test)
#mean probability
tapply(predictTest,test$positive,mean)
0         1
0.4733229 0.9046324
```

Poichè vogliamo una predizione binaria, per convertire le probabilità in una categoria è necessario un valore soglia (*cut-off*) rispetto al quale decidere il risultato, ovvero se appartenente alla classe positiva o negativa. Dato che le due curve di distribuzione di probabilità risultano in parte sovrapposte, sono possibili quattro risultati a seconda della posizione del valore di *cut-off* scelto, schematizzabili uti-

lizzando una tavola di contingenza di dimensione 2x2, detta *confusion matrix*[37]. In questa matrice, in cui le colonne rappresentano la classe reale e le righe la classe predetta, sono possibili quattro differenti classificazioni[38]:

- TP: *true positive*, ovvero il numero di osservazioni correttamente previste positive.
- TN: *true negative*, ovvero il numero di osservazioni correttamente previste negative.
- FP: *false positive*, ovvero il numero di osservazioni erroneamente previste positive (errore di I tipo).
- FN: *false negative*, ovvero il numero di osservazioni erroneamente previste negative (errore di II tipo).

In questo modo, la matrice generata dal livello soglia ci fornisce degli indicatori sintetici che danno informazioni riguardo la capacità previsiva. Una matrice di errori, derivata dalla matrice di confusione generalizzata, consente di valutare quantitativamente le mappe generate utilizzando i modelli di stima dell'area e di confrontarle con le accuratezze ottenute dalle tecniche di classificazione. Nel nostro caso, confrontando le previsioni ottenute con il rendimento reale (osservato), ho ottenuto la seguente matrice di confusione:

```
#confusion matrix
table (test$positive, predictTest>0.5)
FALSE TRUE
   0    97   67
   1    73  840
```

Da questi risultati si possono calcolare tre indicatori statistici: la *sensitivity* (data dal rapporto tra TP ed il totale delle unità previste positive) che misura quanti dati sono stimati correttamente positivi sul totale delle osservazioni effettivamente positive, rispondendo alla domanda "quando il valore effettivo è positivo, quanto spesso la previsione è corretta?"; la *specificity* (data dal rapporto tra TN ed il totale delle unità previste negative) che misura quante unità previste sono realmente negative

[37] Provost, F. J.,et al. (1998), "The case against accuracy estimation for comparing induction algorithms" in *ICML* (Vol. 98, pp. 445-453).

[38] Hay, A. M. (1988), "The derivation of global estimates from a confusion matrix" in *International Journal of Remote Sensing*, 9(8), 1395-1398.

sul totale dei valori osservati negativi, rispondendo alla domanda "quando il valore effettivo è negativo, quanto spesso la previsione è corretta?"; infine l'*accuracy* (rapporto tra i dati previsti correttamente ed il totale delle previsioni) che misura la capacità del modello di prevedere correttamente. L'indice complementare dell'accuratezza, invece, indica l'errore di previsione commesso dal modello[39]. Purtroppo l'operazione di netta divisione tra le due classi, positiva e negativa, operata in base ai termini è di pressocchè impossibile in quanto le *feature* osservate non sono sufficientemente "tipiche" del fenomeno che si vuole distinguere ed è quindi inevitabile introdurre nel classificatore un errore causato dal sovrapporsi nello spazio delle feature delle due popolazioni. Adottando come *threshold* un valore pari a 0.5, si garantisce la massimizzazione della *sensitivity* e della *specificity*, ottimizzando, così, l'accuratezza del modello. Il risultato dell'accuratezza del modello costruito risulta pari 0.8700093. Questo vuol dire che il modello prevede correttamente l'87% dei casi contro il 13% di errore. Quando prevede positivamente ha una probabilità di essere corretto (*sensitivity*) pari all'92%, se invece prevede negativamente avrà un percentuale di correttezza più bassa pari al 59%, contro un errore (1-*specificity*) del 41%, in cui prevede positivamente dati osservati negativi. Per la scelta della soglia, viene in aiuto la curva ROC [40], che grafica i punti di coordinata $(fpr_i; tpr_i)$ come mostrato nella Figura 3.16. La curva ROC (*Receiver Operating Characteristic*) rappresenta un metodo grafico che mostra il *trade off* tra *sensitivity* e *specificity* (in un modo che assuma senso per il nosto modello) per ogni livello della soglia; risulta quindi fondamentale per la valutazione della qualità del classificatore binario. L'asse delle ascisse x corrisponde al valore fpr_i (1-*specificity*) e l'asse delle ordinate y corrisponde al valore tpr_i (*sensitivity*). In altri termini, sull'asse delle ascisse segneremo il valore (1-TN / (TN+FP)) e sulle ordinate il valore TP / (TP+FN). A ciascuna scelta della soglia corrisponderà dunque un punto del diagramma. La curva ROC è la risultante di tutte le possibili scelte del valore soglia, nella figura 3.16 aggiunti ad intervalli di .1. La legenda di colori, invece, mostra come i valori della soglia variano al muoverci sulla curca ROC. L'area sotto la curva (AUC), pari a .8887345, indica la probabilità di classificare correttamente una nuova recensione del prodotto selezionata casualmente[41].

[39] Altman, D. G. e Bland, J. M. (1994), "Diagnostic tests. 1: Sensitivity and specificity" in *BMJ: British Medical Journal*, 308(6943), 1552.

[40] Pencina, M. J. et al.(2008), "Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond" in *Statistics in medicine*, 27(2), 157-172.

[41] Bradley, A. P. (1997), "The use of the area under the ROC curve in the evaluation of machine learning algorithms" in *Pattern recognition*, 30(7), 1145-1159.

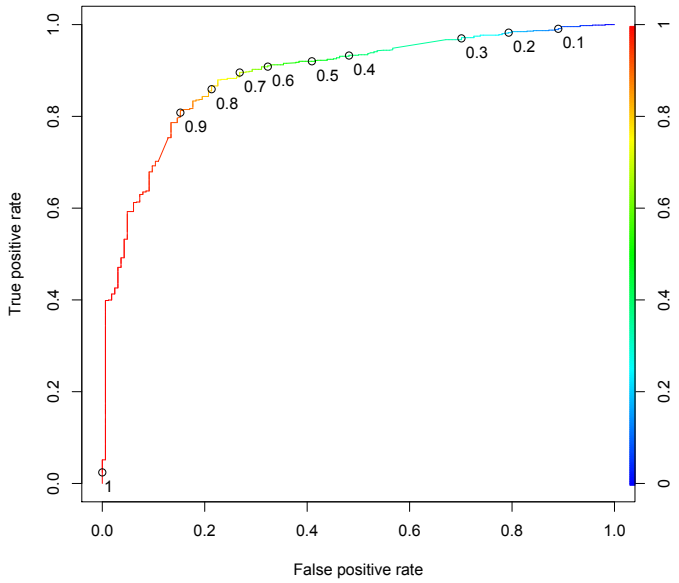

Figura 3.16. Curva ROC

Non è detto che classificare bene gli oggetti presenti nel *training set* implichi una buona capacità di generalizzazione. Difatti, talvolta si può verificare il caso di *overtraining*: il sistema ha imparato talmente bene i *pattern* del *training set* che non è più in grado di generalizzare[42]. Un valore di accuratezza così alto, però, suggerisce come sul *training set* non si sia verificato alcun problema dell'*overfitting*. Predicendo sul set di dati di allenamento e selezionando come soglia 0.5, la *confusion matrix* generata è la seguente:

```
#testing overfitting
predictTrain = predict(logReg,type="response",newdata=train)
table(train$positive,predictTrain>0.5)
     FALSE TRUE
  0    271  113
  1    106 2023
```

L'accuratezza risulta, difatti, pari a 0.9128532. Poichè costruendo il modello di regressione logistica, il nostro obiettivo è ottenere un classificatore più accurato

[42] Tetko, I. V. et al. (1995), "Neural network studies. 1. Comparison of overfitting and overtraining" in *Journal of chemical information and computer sciences*, 35(5), 826-833.

rispetto alla *baseline*, calcoliamo l'accuratezza di quest'ultima, che fornisce come previsione sempre il valore più frequente.

```
#baseline
table(test$positive)
  0   1
164 913
```

L'accuratezza della baseline è pari a 0.8477252. Di conseguenza, il modello di regressione logistica costruito fornisce prestazioni di classificazione più elevate.

3.4.2 Alberi di Classificazione e Regressione

Gli alberi decisionali costituiscono il modo più semplice di classificare un set di osservazioni in un numero finito di classi. Questo è il motivo per cui, per predire se una recensione sarà positiva o no, ho applicato l'algoritmo CART (*Classification and Regression Trees*), oggetto di discussione nella seguente sezione. Sia chiaro come l'obiettivo perseguito con il CART coincide con quello proposto con il modello di regressione logistica illustrato nella sezione 3.4.1: valutare l'accuratezza e le capacità di generalizzazione del classificatore ed operare un confronto tra l'algoritmo, la *baseline* ed altri *classifier*. Semplicemente, a differenza del modello di regressione logistica, il CART per come è costruito facilita la lettura delle predizioni.

Gli alberi di classificazione e regressione sono metodi di *machine learning* applicati su set di dati e volti alla costruzione di modelli predittivi[43]. L'algoritmo funziona partizionando lo spazio dei dati in modo ricorsivo e binario e, all'interno di ogni partizione, adatta un semplice modello di predizione. In particolare, il partizionamento avviene suddividendo ripetutamente le osservazioni in sottoinsiemi omogenei rispetto alla variabile dipendente. La suddivisione produce una gerarchia ad albero, dove i nodi sono i *subset* di osservazioni etichettati con il nome delle variabili indipendenti, gli archi (i rami dell'albero) forniscono le regole di *splitting* binario e sono etichettati con i possibili valori che la variabile dipendente può assumere. Le foglie, infine, sono etichettate con le differenti modalità della variabile risposta che descrivono le classi di appartenenza[44]. Un oggetto, in pratica, è classificato seguendo un percorso di *splitting* che porta dalla radice (primo nodo) ad una foglia.

[43]Robert, C. (2014), "Machine learning, a probabilistic perspective".

[44]Loh, W. Y. (2011), "Classification and regression trees" in *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1), 14-23.

In questo modo, il partizionamento può essere rappresentato graficamente come un albero di classificazione dicotomica, come mostrato nella Figura 3.17.

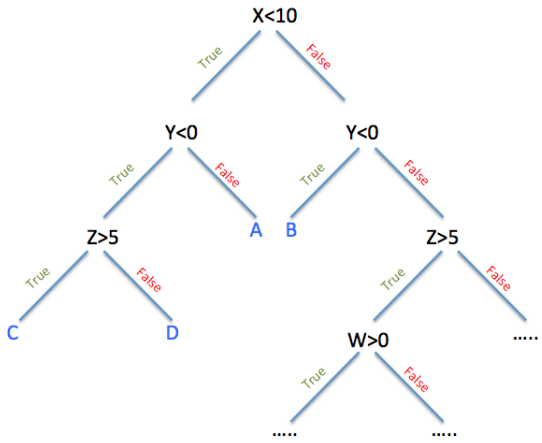

Figura 3.17. Esempio di decision tree

L'algoritmo, reso popolare da Breiman[45], è un esempio di processo decisionale multifase: invece di utilizzare tutte le variabili contenute nel *dataset* congiuntamente per creare una regola di decisione, ne utilizza differenti sottoinsiemi in corrispondenza di diversi livelli dell'albero. Gli alberi di classificazione sono progettati per le variabili dipendenti che accettano un numero finito di valori non ordinati, con l'errore di predizione misurato in termini di costo di errata classificazione. Gli alberi di regressione, invece, sono utilizzati per variabili dipendenti che assumono valori discreti continui o ordinati, con un'errore di predizione tipicamente misurato dalla differenza quadratica tra i valori osservati e quelli previsti. La costruzione di un *decision tree* include tre passi[46]:

1 Selezione di una regola di *splitting* per ogni nodo; ciò significa determinare le variabili, insieme al rispettivo valore soglia, che saranno usate per partizionare il *dataset* ad ogni nodo.

2 Determinare quali nodi sono da intendersi terminali; quindi per ogni nodo bisogna decidere quando continuare con gli *split* e quando invece fermarsi e considerare il nodo come terminale (e di conseguenza assegnargli un'etichetta).

[45] Breiman, L. (1996), "Bagging predictors" in *Machine learning*, 24(2), 123-140.

[46] Rao V. (2013), "Introduction to Classification & Regression Trees (CART)". Accessibile su: *https://www.datasciencecentral.com/profiles/blogs/introduction-to-classification-regression-trees-cart*.

3 Assegnare le etichette ad ogni nodo terminale, ad esempio minimizzando il valore atteso di errata classificazione.

In riferimento al punto 2, senza un'adeguata regola di *splitting*, si corre il rischio di costruire alberi troppo grandi con ridotta capacità di generalizzazione, oppure alberi troppo piccoli che invece approssimano male i dati (*overfitting*). In situazioni del genere, lasciar crescere l'albero senza stabilire un limite di qualsiasi natura può far si che l'albero ottenuto risulti non interpretabile e crei un numero elevato di regole, diventando di fatto non predittivo. Esistono, quindi, dei criteri di controllo che limitano la crescita degli alberi. In tale ambito rientra anche la fase di *pruning* che consiste nell'ottenere da un albero il più piccolo sottoalbero, che di fatto non comprometta l'accuratezza della classificazione/previsione resa possibile dall'albero originario. L'algoritmo CART effettua la fase di *pruning* semplicemente verificando se il miglioramento nell'accuratezza giustifica la presenza aggiuntiva di altri nodi[47].

L'obiettivo, quindi, è stato costruire un albero decisionale splittando i valori delle variabile indipendente, senza fare assunzioni di linearità. In questo modo, il modello produce una serie di regole decisionali che possono essere interpretate in modo semplice ed intuitivo. Come variabile dipendente dicotomica è stato utilizzato il vettore Positive contenuto nel dataset, costituito da TRUE se l'afinn score della recensione è maggiore di 0 e FALSE nel caso opposto. Come fatto nella sezione 3.4.1, per allenare il classificatore probabilistico si è diviso il vettore Positive in due *subset* in base alla ratio 7:3 in modo da costruire un *training set* come dimensione di allenamento ed un *validation set* per poi verificarne le capacità discriminative. Per performare il modello CART, sono state necessarie funzioni dei pacchetti rpart e rpart.plot. In particolare, utilizzando la funzione rpart() è stato costruito il modello sul *training set*, come mostrato nella Figura 3.18. In questo modello, il vettore Positive rappresenta la variabile dipendente, mentre i 112 termini rappresentano le variabili indipendenti. Si noti che il *method argument* inserito nella funzione rpart come dato di input è *class*; questo perchè la variabile risposta è categorica.

```
1 CART = rpart(Positive ~., data=trainset, method='class')
```

[47] Steinberg, D. e Colla, P. (2009) "The top ten algorithms in data mining", 9, 179.

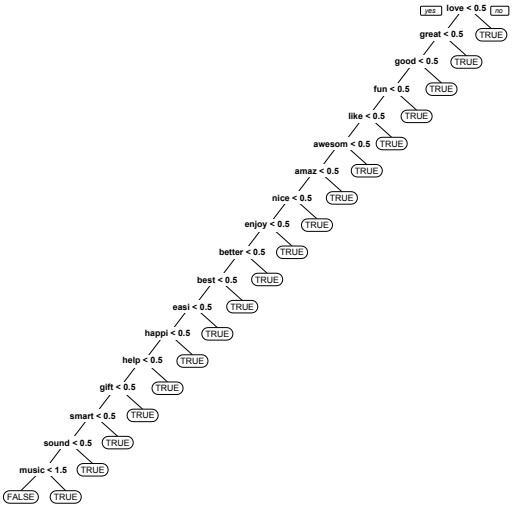

Figura 3.18. Albero decisionale

Come leggere l'albero? Per predire il nostro valore output per una nuova osservazione, possiamo seguire gli *split* lungo i vari nodi dell'albero fino al raggiungimento dell'elemento foglia. L'albero decisionale mostra la gerarchia di decisioni e le conseguenze in base alla frequenza di occorrenza dei termini. Nel caso in esame, se il termine "*love*" è maggiore di 0.5, allora la recensione verrà classificata come positiva; in caso contrario, dobbiamo seguire lo *split* fino al termine "*great*". Se "*great*" non è minore di 0.5, possiamo classificare la recensione come positiva; in caso contrario, sarà necessario seguire lo *split* seguente. Questa logica è interata fino all'ultimo *split*. Il senso è, intuitivamente, il seguente: se il termine "*love*" occorre in una recensione, è più probabile che la stessa sia positiva; se "*love*" non appare ma il termine "*great*" è presente, la recensione sarà probabilmente positiva. Il passo successivo è stato verificare la bontà delle predizioni fatte dal modello sul *testing set*. Per questo motivo è stato generato un vettore di predizioni utilizzando la funzione predict() applicata sui dati del *validation set*; il risultato è stato una matrice di confusione che, come spiegato nella sezione 3.4.1 rappresenta una tabella riepilogativa dei risultati di previsione (corretti o meno) in grado di descrivere le prestazioni del classificatore sui dati del *testing set* (per i quali sono noti i valori reali). Ogni riga della matrice di confusione rappresenta le istanze di una classe effettiva e ogni colonna rappresenta le istanze di una classe prevista. Il numero di previsioni corrette e non corrette viene riepilogato con valori di conteggio e suddiviso per ogni classe. Nel caso specifico, la *confusion matrix* generata è stata la seguente:

```
        FALSE TRUE
FALSE   109    55
TRUE     68   845
```

L'accuratezza del modello costruito è risultata pari a .8857939. Poichè il nostro obiettivo è disporre di un classificatore più accurato rispetto alla *baseline*, calcoliamo l'accuratezza di quest'ultima che, ricordiamo, fornisce come previsione sempre il valore più frequente.

```
#baseline
table(testset$Positive)
FALSE  TRUE
  164   913
> 913/(164+913)
[1] 0.8477252
```

L'accuratezza della *baseline* risulta pari a 0.8477252 e quella del classificatore logistico pari a 0.8700093; di conseguenza, il CART costruito fornisce prestazioni di classificazione più elevate. Spesso possono presentarsi alcuni problemi nella costruzione di alberi di classificazione binari e nella stima del costo di errata classificazione ad essi associato. Infatti, dopo aver individuato un albero di grandi dimensioni, ci si riconduce a un sottoalbero ottimale di dimensione ridotta attraverso il *pruning*, effettuato con l'intento di pervenire ad una struttura che al tempo stesso sia di facile interpretazione e che minimizzi il costo di errata classificazione[48]. Per stimare il costo di errata classificazione, come abbiamo visto, la *cross-validation* sembra la scelta più appropriata, anche se non è del tutto immune dagli inconvenienti. Per rimediare ad alcuni di questi incovenienti nei problemi di classificazione, in genere, si utilizza la *k*-fold cross validation stratificata, tecnica oggetto di discussione nella sezione 3.4.3.

3.4.3 *K*-fold Cross Validation Technique

Il principale problema dei modelli di *machine learning* è che non si sa come performano fino a quando le loro prestazioni non verranno testate su un set di dati indipendente (*testing set*), ovvero differente da quello impiegato per addestrare

[48]Esposito, F. et al. (1997), "A comparative analysis of methods for pruning decision trees" in *IEEE transactions on pattern analysis and machine intelligence*, 19(5), 476-491.

il modello. La misura standard di accuratezza per i modelli di apprendimento è l'errore di previsione (PE), ovvero la "perdita" prevista per i casi futuri. Questi algoritmi sono spesso confrontati in base alla loro prestazione media, formalmente definita come il valore atteso dell'errore di previsione (EPE) rispetto al *training set*. Quando però la distribuzione dei dati è sconosciuta, PE ed EPE non possono essere calcolati. Se la quantità di dati è abbastanza grande, il PE può essere stimato dall'errore medio generato su un *testing set*. Le normali stime della varianza per mezzo di set di dati indipendenti possono essere calcolate per ricavare l'errore di previsione stimato e per valutare la significatività statistica delle differenze tra i modelli. In questa situazione, per stimare il PE o EPE si utilizzano metodi di ricampionamento intensivi come, per esempio, la *cross-validation*. Mentre è noto che la *cross-validation* fornisce una stima obiettiva dell'EPE, è anche noto che la sua varianza può essere molto ampia[49]. Questa varianza dovrebbe essere stimata per fornire intervalli di confidenza fedeli su PE o EPE e per verificare il significato delle differenze osservate tra gli algoritmi [50]. Così, per ridurre la varianza, vengono eseguiti diversi cicli di convalida incrociata utilizzando diversi *training* e *testing set*. La tecnica, detta *k*-fold cross validation, è ampiamente utilizzata per valutare le prestazioni di un modello predittivo costruito su un set di dati di addestramento stimando, sul *testing set*, quanto le previsioni fatte dal modello sono risultate accurate. Nella tipica cross-validation, il *training* ed il *testing set* devono essere incrociati in *round* successivi in modo tale che ogni punto di dati abbia una possibilità di essere convalidato. Per i problemi di classificazione, in genere, si utilizza la *k*-fold cross validation stratificata, in cui il partizionamento prevede che il campione originario di dati sia suddiviso in *k* sottocampioni di egual numerosità, selezionati in modo che ognuno di questi contenga all'incirca le stesse proporzioni delle etichette di classe. Di questi *k* sottoinsiemi, il *k*-esimo viene utilizzato come *testing set* per testare il modello, mentre i rimanenti *(k-1)* sottoinsiemi vengono uniti ed impiegati come *training set*. Il processo di cross-validation viene ripetuto *k* volte ed ognuno dei *k* sottoinsiemi viene utilizzato esattamente una volta come *testing set*, come illustrato nella Figura 3.19.

Come è possibile notare, ogni *fold* può trovarsi in un set di convalida esattamente una volta e può essere inserito in un set di allenamento *(k-1)* volte. Questo riduce in modo significativo il *bias* di campionamento asimettrico tipico della suddivisio-

[49] Breiman, L. (1996), "Bagging predictors" in *Machine learning*, 24(2), 123-140.

[50] Bengio, Y. e Grandvalet, Y. (2004), "No unbiased estimator of the variance of k-fold cross-validation" in *Journal of machine learning research*, 5(Sep), 1089-1105.

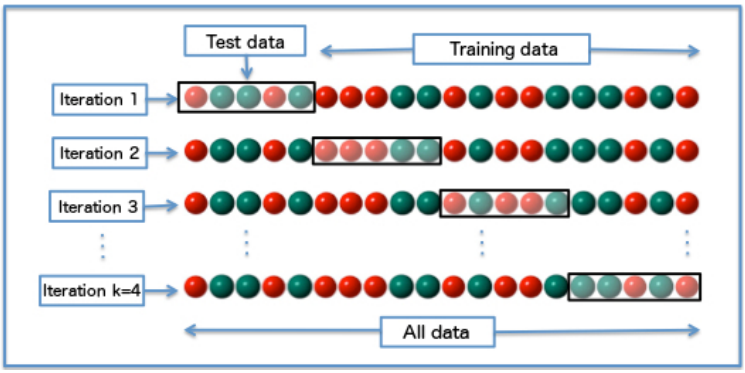

Figura 3.19. Processo di *k*-fold cross validation

ne del *dataset* in due sole parti (*training* e *testing set*); in più elimina il problema dell'*overfitting* e minimizza, in modo significativo, la varianza visto che i dati vengono utilizzati anche nel testing set[51]. Per la stima dell'accuratezza del modello come valore si utilizza la media dell'accuratezza ottenuta da tutti i *k* processi di cross-validation. Come regola generale ed evidenza empirica, di solito si utilizza un numero di *fold* pari a 10, ma questa non è una regola rigida, quindi *k* può assumere qualsiasi valore. Le dimensioni degli alberi decisionali spesso modificano l'accuratezza della predizione. In generale, più grande è l'albero maggiore sarà l'accuratezza del modello, ma se l'albero è troppo grande può verificarsi un problema di *overfitting* sul *training set* e ciò si traduce in minor accuratezza e robustezza. Difatti il CART, di solito, si adatta in modo eccessivo al modello, creando un albero che spiega sostanzialmente tutta la devianza presente nei dati originali in un modo troppo specifico e troppo costruito sul *training set*. È quindi necessario ridimensionare l'albero (tecnica di *pruning*) ad un livello in cui ci si può ragionevolmente aspettare che lo stesso risulti robusto. In questo senso, è possibile selezionare un albero più piccolo se il metodo di cross validation indica che, sebbene possa essere ridotta un'ulteriore devianza, l'entità della riduzione non giustifica un albero troppo complesso. Un metodo comune utilizzato per il *pruning* è, appunto, la *k*-fold cross validation.

Nel caso specifico, la *k*-fold cross validation è stata applicata per testare l'accuratezza del modello CART costruito nella sezione 3.4.2. In primo luogo, è necessario l'utilizzo di alcune funzioni disponibili nei pacchetti caret e e1071. Come primo step, ho diviso il set di dati in *k*=10 *fold* costituiti dallo stesso numero di parole.

[51]Mullin, M. D. e Sukthankar, R. (2000, June), "Complete Cross-Validation for Nearest Neighbor Classifiers" in *ICML* (pp. 639-646).

Come sappiamo l'algoritmo opera escludendo iterativamente un gruppo alla volta e cerca di predirlo con i gruppi non esclusi; ciò al fine di verificare la bontà del modello di predizione utilizzato. Gli alberi vengono generati per *(k-1)*=9 *fold* e convalidati rispetto al *k*-esimo, con la minima devianza media che indica l'albero dalle dimensioni migliori. L'output della *k*-fold cross validation è il *complexity parameter*(cp) che misura il *trade-off* tra complessità del modello ed accuratezza sul *training set*. In particolare, il cp viene utilizzato per controllare la dimensione dell'albero decisionale e per selezionare la dimensione di quello ottimale, che si traduce in definire il livello di *pruning*. Se il costo di aggiungere un'altra variabile all'albero decisionale dal nodo corrente è superiore al valore del cp, la costruzione dell'albero non continua; in questo senso il cp definisce il numero di *split* evitando qualsiasi forma di *data overfitting*. La convenzione è di avere un albero piccolo e quello con il minor *cross-validated error*. Dopo aver scelto il numero di fold (10), applicando la funzione trainControl(), ho definito l'intervallo di valori per cp che deve essere testato e poi verificato utilizzando la funzione expandGrid(): tutti i valori da 0.1 a 0.5 con incrementi di 0.01. In seguito la funzione train() che fornisce il livello ottimale di *pruning* in base ai valori del cp.

```
numFolds = trainControl( method = "cv", number = 10 )
cpGrid = expand.grid( .cp = seq (0.01,0.5,0.01))
train(positive~.,trainset,"rpart",trControl=numFolds,tuneGrid=cpGrid)
```

In particolare, la funzione train() ha restituito un vettore di valori del cp con *accuracy* associata. Poichè ad un cp pari a .01 è associato il livello *accuracy* massimo (0.9060754), ho costruito un nuovo modello CART inserendo come dato di input alla funzione rpart() un cp pari a 0.01.

```
CART2 = rpart(Positive ~., data=trainset, method="class", cp=0.01)
predictCART2 = predict(CART2, newdata=testset, type="class")
table(testset$Positive, predictCART2)
        predictCART2
                FALSE TRUE
  FALSE     109        555
  TRUE       68         845
```

Verificando la predizione sul *testing set*, il livello di accuratezza del modello calcolato sulla matrice di confusione è stato pari a 0.8857939. La Figura 3.20 fornisce i risultati della cross-validation, mostrando i valori dei cp associati ai rispettivi errori. I valori

cp sono tracciati rispetto alla media geometrica per rappresentare la deviazione fino al raggiungimento del valore minimo. Il valore del cp ottimale (appunto 0.01) è quello cui corrisponde l'errore minimo. Nel caso particolare, al livello del cp che minimizza l'errore l'algoritmo opera un *pruning* dell'albero, che definisce una *size of tree* ottimale di 19 *split*.

Figura 3.20. Complexity parameter

In sintesi, l'algoritmo passa attraverso ogni albero e calcola il suo tasso di errore sul *testing set* che è stato tenuto fuori. Analizza tutti i parametri di complessità e determina come l'albero avrebbe performato sul *testing set* se fosse stato usato quel determinato cp. Concludendo, se confrontiamo l'accuratezza del CART costruito nella sezione 3.4.2, pari a 0.8857939, con l'accuratezza del modello costruito con un cp pari a 0.01, il risultato è il medesimo. Ciò equivale a dire che il modello CART originario non presenta alcun problema di *overfitting* sui dati.

3.4.4 Random Forest Classifier

Un classificatore in grado di migliorare l'accuratezza del modello CART è il Random Forest Classifier (o *Bootstrap Aggregation*), oggetto di spiegazione nella seguente sezione. Un problema del modello CART è che sceglie quale variabile splittare utilizzando un algoritmo che minimizza l'errore. Così facendo, il risultato è una serie di alberi decisionali accomunati da molte somiglianze strutturali, nonchè un'alta correlazione tra le rispettive previsioni. La soluzione arriva dai metodi di *ensemble learning*. Questi metodi impiegano più modelli di apprendimento per ottenere risultati predittivi migliori: il Random Forest[52] in particolare, crea

[52]Breiman, L. (1996), "Bagging predictors" in *Machine learning*, 24(2), 123-140.

un'intera foresta di alberi decisionali casuali non correlati per ottenere la migliore risposta possibile. Così facendo, tale classifcatore di *ensemble learning* si prefigura come in grado di modificare l'algoritmo classico: gli alberi secondari vengono appresi facendo in modo che le predizioni risultanti da tutti i sottoalberi abbiano minor correlazione possibile. In generale, combinare le previsioni di più modelli in *ensemble* funziona meglio se le previsioni dei sottomodelli non sono correlate (o lo sono debolmente).

Per definizione, il Random Forest è un metodo di *ensemble learning* che opera costruendo una moltitudine di alberi decisionali, ognuno dei quali sviluppato da un *bootstrap sample* estratto dai dati di *training*[53]. In particolare, ogni albero produce una risposta in base ad un complesso di variabili indipendenti. Sia il numero di alberi nella foresta che il numero di variabili nel sottoinsieme sono *hyper-parameters*, quindi scelti a priori. Come funziona? Consideriamo di disporre di una popolazione composta da 1000 osservazioni per 10 variabili. Quello che il Random Forest fa è creare più modelli CART con differenti combinazioni di campioni e variabili iniziali. Ad esempio, se per costruire un modello CART è necessario un campione casuale di 100 osservazioni e 5 variabili iniziali randomicamente scelti, l'algoritmo Random Forest ripeterà il processo di selezione casuale (quindi differenti regole di *splitting*) 10 volte effettuando una previsione finale su ogni osservazione. Ogni albero voterà un *outcome* e quello che riceverà la maggioranza dei voti verrà selezionato. La previsione finale è funzione delle previsioni fatte e può semplicemente essere rappresentata dalla media di ogni previsione. L'utilizzo di un complesso di alberi può portare a significativi miglioramenti nell'accuratezza della previsione, ovvero ad una migliore capacità di predire nuovi casi di dati. Per questo motivo, il Random Forest viene applicato per costruire modelli predittivi sia nei casi di classificazione che in quelli di regressione. Per i problemi di classificazione, la risposta prodotta consiste nell'etichettare l'appartenenza ad una classe, associando (o classificando) un insieme casuale di predittori ad una delle categorie presenti nella variabile dipendente. L'algoritmo definisce una funzione margine che misura di quanto il numero medio di voti per la classe corretta supera il voto medio per qualsiasi altra classe presente nella variabile dipendente. In pratica, come mostrato nella Figura 3.21, ogni albero esprime un voto e la classe più popolare viene votata. Questa misura ci fornisce non solo un modo conveniente per fare previsioni, ma anche un modo di associare una misura di confidenza con tali previsioni. Per i problemi di regressione, invece, dato un set di predittori, la risposta dell'albero è una stima

[53]Statsoft (2018), "Random Forest". Accessibile da: *http://www.statsoft.com/Textbook/Random-Forest*.

della variabile dipendente. La risposta di ogni albero dipende dal set di predittori, selezionato casualmente dalla distribuzione stessa e per ognuno degli alberi.

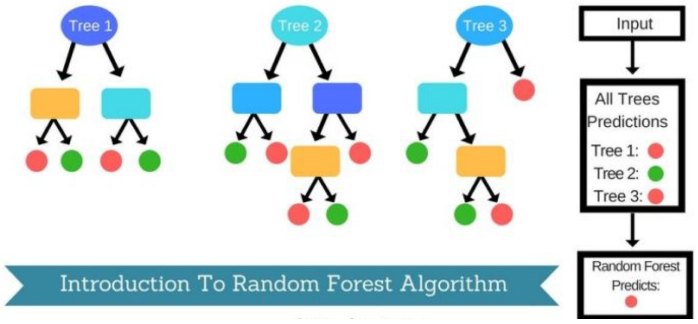

Figura 3.21. Random Forest

Il Random Forest viene generalmente applicato quando si dispone di un elevato numero di variabili predittive e di una rilevante dimensione campionaria (come nel nostro caso). L'algoritmo è in grado di catturare la varianza di più variabili di input contemporaneamente e permette, per determinare la previsione, l'inserimento in input di un elevato numero di osservazioni. Il Random Forest, oltre a gestire le complicazioni causate dall'elevata dimensionalità dei dati, può gestire con successo anche i problemi derivanti da multicollinearità; questo perchè, a differenza del CART, risulta veloce ed insensibile ai problemi di *overfitting*. Passando al caso pratico, per applicare l'algoritmo sul mio set di dati di *training* (nominato "trainset"), ho installato ed utilizzato le funzioni disponibili nel pacchetto randomForest. Utilizzando la funzione randomForest(), ho costruito il modello sul *training set* generando 200 differenti alberi.

```
randomForest = randomForest(Positive ~., data = trainset, ntree = 200)
```

Lo step successivo è stato verificare sul *testing set* la bontà delle predizioni. Per questo motivo, ho generato un vettore delle predizioni utilizzando la funzione predict() da applicare sui dati del *validation set*. Predicendo, il risultato è stata una matrice di confusione con accuratezza complessiva pari a .8839369.

```
       FALSE  TRUE
FALSE    87     77
TRUE     48     865
```

Poichè costruendo il modello Random Forest il nostro obiettivo è ottenere un classificatore più accurato rispetto alla *baseline*, confrontiamo ora l'accuratezza dei due modelli. Come visto nella sezione 3.4.2, la *baseline* assicura un'accuratezza pari a 0.8477252, il Random Forest, invece, pari a 0.8839369. Quest'ultimo, di conseguenza, fornisce prestazioni di classificazione più elevate. Il confronto più interessante, però, è con il modello CART. Se l'obiettivo è predire, il Random Forest *classifier* è generalmente un modello migliore del CART in quanto corregge i problemi di *overfitting* del *decision tree* sul *training set* e migliora la capacità di generalizzazione del modello. L'idea è che maggiore è il numero di alberi, maggiore sarà la precisione del modello. In questo caso, però, comparando l'acuratezza dei due modelli, notiamo come la differenza sia minima, pressocchè nulla: l'accuratezza del modello CART è pari a 0.8857939 quella del Random Forest pari a 0.8839369 La scelta del modello da adottare, di natura politica, richiede di ponderare il *trade-off* tra accuratezza e complessità del modello. Per tale ragione, in questo caso è preferibile il modello CART, poichè più semplice ed immediato da interpretare.

Conclusioni

Il lavoro di tesi ha seguito un approccio quantitativo nella guida all'analisi testuale ed ha esaminato tutti i risultati rispetto alle domande di ricerca, inerenti al fornire strumenti per la raccolta automatica di dati testuali che permettano alle aziende di monitorare in *real-time* la propria *web reputation* e la *brand perception*. La scelta di utilizzare come metodo di ricerca la raccolta automatizzata dei dati testuali tramite un linguaggio di programmazione consente di superare le limitazioni tipiche di un sondaggio, attività dispendiosa in termini di tempo ed accessibilità, con risultati non sempre accurati.
In questa ricerca è stato presentato un approccio all'analisi testuale a livello di *topic* che raggruppa le parole più frequenti in cluster associati a differenti *topic* discussi. I metodi di Topic Modeling e clustering hanno permesso, seguendo approcci differenti, di comprendere e sintetizzare la raccolta testuale fornendo una finestra pubblica su quali sono gli aspetti del prodotto di maggior interesse per i consumatori: l'utilizzo domestico del prodotto, la capacità del *device* di riprodurre musica, la qualità del prodotto e la semplicità ed il piacere derivanti dall'utilizzo dello *smart speaker*. In particolare, poichè i *topic* utilizzati nel linguaggio naturale ammettono *overlapping* di parole, il Topic Modeling ha superato le limitazioni tipiche dell'*hard clustering*. Scoprire *topic* latenti e regole di associazione all'interno della collezione ha consentito di affrontare il problema della categorizzazione della polarità del *sentiment* e fornire *text classifier* predittivi.
Dopo aver identificato, quantificato e caratterizzato il contenuto sentimentale dell'unità di testo (*overall polarity*), la conclusione raggiunta è che il *sentiment* che i recensori mostrano nei confronti del prodotto è decisamente positivo; questo potrebbe voler dire che le attese che i consumatori avevavo non sono state disattese dopo l'acquisto. Il confronto tra i risultati ottenuti con tre differenti lessici ci ha fornito una prospettiva a più ampio raggio per meglio comprendere l'orientamento emozionale delle recensioni. Infine è stato presentato un metodo per la collezione automatica di un *text corpus* utilizzabile per allenare *sentiment classifier*, in grado

di identificare automaticamente la polarità per una nuova recensione raggiungendo risultati significativamente migliori rispetto alla *baseline*. Si è constatato che i classificatori di *sentiment* dipendono fortemente da domini o argomenti e nessuno dei modelli di classificazione supera in modo consistente l'altro. Si è comunque scoperto che il modello CART garantisce prestazioni più elevate rispetto al modello di regressione logistica e al Random Forest.
Nelle analisi fatte, abbiamo adottato come misura di similarità la frequenza di occorrenza dei termini. L'analisi delle relazioni basate sulla distanza ovviamente ha dei limiti. Ad esempio, anche quando un termine soggetto e una *sentiment word* sono contenuti nella stessa frase e posti vicini l'uno all'altro, il termine soggetto e il termine sentimento potrebbero non essere affatto correlati. La maggior parte delle applicazioni di *sentiment analysis* mira a classificare l'intero documento in positivo o negativo presupponendo che, ad essere oggetto di tutte le espressioni di sentimento sia il soggetto della raccola di recensioni. in realtà, espressioni che violano questa ipotesi confondono il giudizio di classificazione. Al contrario, analizzando le relazioni tra espressioni di sentimento e soggetti, è possibile fare analisi più approfondite e precise. In quest'ottica, il compito di uno studio successivo potrebbe essere analizzare frammenti di testo in modo da identificare relazioni semantiche tra soggetti ed espressioni di *sentiment* a loro relative; in questo caso, la polarità del *sentiment* potrebbe risultare completamente diversa a seconda delle relazioni.

R commands

```
#data scraping
library(rvest)
ASIN_code = "B06XCM9LJ4"
#funzione sorgente
source("https://raw.githubusercontent.com/rjsaito/Just-R-Things
+/master/Text%20Mining/amazonscraper.R")
#creating the dataframe reviews
dataset=NULL
pages=2341
#forloop
for(page_num in 1:pages){
#concatenare l'URL con l'ASIN and ed il numero di pagina
url = paste0("http://www.amazon.com/product-reviews/",ASIN_code,
+"/?pageNumber=", page_num)
#lettura dell'HTML dell'url
doc = read_html(url)
#parsing dell'html
reviews = amazon_scraper(doc, reviewer = F, delay = 2)
dataset = rbind(dataset,reviews)
}

options(stringsAsFactors=F)
#cancellare variabili non necessarie
dataset$X=NULL
dataset$title=NULL
dataset$author=NULL
dataset$date=NULL
dataset$ver.purchase=NULL
dataset$format=NULL
dataset$helpful=NULL
dataset$X=NULL

#numero di caratteri in ogni elemento del vettore comments
nchar = nchar(dataset$comments)
```

```
nchar = as.data.frame(nchar)
#combinare dataset e nchar
dataset = cbind(dataset,nchar)
sum(dataset$nchar)
[1] 519413
#media
sum(dataset$nchar)/3590
[1] 144.6833
#deviazione standard
sd(dataset$nchar)
[1] 271.2767
#mediana
median(dataset$nchar)
[1] 72

#distribuzione di frequenze di nchar
library(ggplot2)
ggplot(nchar, aes(x = nchar)) +
geom_histogram(aes(fill = ..count..), binwidth = 25) +
scale_x_continuous(breaks = seq(0, 5500, 1000),
limits=c(0, 5500)) +
scale_y_continuous(name = "Count")+
geom_vline(xintercept = 145, size = 0.5, colour = "#FF3721",
linetype = "dashed")

#media e sd del vettore stars
sum(dataset$stars)/3590
[1] 4.430919
sd(dataset$stars)
[1] 1.076144
#distribuzione di frequenze del vettore stars
library(ggpubr)
ggplot(dataset, aes(stars)) +
geom_bar(fill = "#0073C2FF") +
theme_pubclean()

#creare e pulire il corpus delle recensioni
library(tm)
library(SnowballC)
corpus = VCorpus(VectorSource(dataset$comments))
#convertire in lowercase
corpus = tm_map(corpus, content_transformer(tolower))
#rimuovere punteggiatura
corpus = tm_map(corpus, removePunctuation)
```

```
#rimuovere numeri
corpus = tm_map(corpus, removeNumbers)
#rimuovere stopwords
corpus = tm_map(corpus, removeWords, stopwords("english"))
#wordcloud
library(wordcloud)
wordcloud(corpus,max.words=300,random.color=T,min.freq=30,
+colors=rainbow(50))
#stemming
corpus = tm_map(corpus, stemDocument)
tdm = TermDocumentMatrix(corpus)

#Spearman rank correlation
corr = cor.test(dataset$nchar,dataset$stars,method="spearman")

  Spearman's rank correlation rho

data:  dataset$nchar and dataset$stars
S = 9821800000, p-value < 2.2e-16
alternative hypothesis: true rho is not equal to 0
sample estimates:
       rho
-0.2736798

#Topic Modeling Method
library(tidytext)
library(topicmodels)
library(ggplot2)
library(dplyr)
library(bindrcpp)
#matrice dtm
dtm = DocumentTermMatrix(corpus)
rowTotals = apply(dtm , 1, sum)
dtm = dtm[rowTotals> 0,]
#lda in 4 topic
lda = LDA(dtm, k = 4, control = list(seed = 1154))

#topics
topics = tidy(lda, matrix = "beta")
topics2=tidy(lda, matrix="gamma")
#top terms
top.terms = topics %>%
group_by(topic) %>%
top_n(10, beta) %>%
```

```
ungroup() %>%
arrange(topic, -beta)
#plotting
top.terms %>%
mutate(term = reorder(term, beta)) %>%
ggplot(aes(term, beta, fill = factor(topic))) +
geom_col(show.legend = FALSE) +
facet_wrap(~ topic, scales = "free") +
coord_flip()

#clustering
#rimuovere termini sparsi
freqTerms = removeSparseTerms(tdm, 0.95)
freqTerms.df = as.data.frame(as.matrix(freqTerms))
library(cluster)
data.df.scale = scale(freqTerms.df)
#matrice delle distanze
dist = dist(data.df.scale, method = "euclidean")
#clustering gerarchico
hcluster = hclust(dist, method="ward.D")
#taglio in 4 cluster
hclusterID = cutree(hcluster, k=4)
#dendogramma
labelColors = c("darkgrey", "blue", "green","yellow")
# function to get color labels
hcd = as.dendrogram(hcluster)
colLab = function(n) {
if (is.leaf(n)) {
a = attributes(n)
labCol = labelColors[hclusterID[which(names(hclusterID) == a$label)]]
attr(n, "nodePar") = c(a$nodePar, lab.col = labCol)
}
n}
dendrogram = dendrapply(hcd, colLab)
plot(dendrogram, main = "Hcluster Dendrogram", type = "rectangle")
rect.hclust(hcluster, k=4, border="red")

#k-mean clustering con 4 centroidi
kmeans4 = kmeans(dist, centers=4, nstart=100)
clusplot(as.matrix(dist), kmeans4$cluster, color=T, shade=T,
+labels=3, lines=0, main="kmeans clusplot")

#lista delle sentiment words
library(tidytext)
```

```
167 library(dplyr)
168 tidy = tidy(tdm)
169 sentwords = tidy%>%
170 inner_join(get_sentiments("bing"), by = c(term = "word"))
171 #partizione tra parole positive e negative
172 table(sentwords$sentiment)
173 negative positive
174     1061      5827
175
176 #top sentwords
177 library(ggplot2)
178 sentwords %>%
179   count(sentiment, term, wt = count) %>%
180   ungroup() %>%
181   filter(n >= 25) %>%
182   mutate(n = ifelse(sentiment == "negative", -n, n)) %>%
183   mutate(term = reorder(term, n)) %>%
184   ggplot(aes(term, n, fill = sentiment)) +
185   geom_bar(stat = "identity") +
186   ylab("Contribution to sentiment") +
187   coord_flip()
188
189 #polarity del sentiment
190 library(syuzhet)
191 #punteggio afinn
192 afinn = get_sentiment(dataset$comments,method="afinn")#punteggio
193 afinn = as.data.frame(afinn)
194 sum(afinn)
195 [1] 15044
196 #punteggio bing
197 bing = get_sentiment(dataset$comments,method="bing")#punteggi
198 bing = as.data.frame(bing)
199 sum(bing)
200 [1] 5999
201 #nrc
202 nrc = get_nrc_sentiment(dataset$comments)
203 barplot(sort(colSums(prop.table(nrc[, 1:8]))),horiz = TRUE,
204 +cex.names = 0.7, las = 1, main = "Emotions in reviews",
205 + xlab="Percentage",col=rainbow(8))
206 #pie chart ratio tra recensioni positive e negative
207 dataset=cbind(dataset,afinn)
208 dataset$Positive = as.factor(dataset$afinn>0)
209 pos = gsub("TRUE","positive",dataset$Positive)
210 pos = as.data.frame(pos)
```

```
pos = gsub("FALSE","negative",pos$pos)
pos = as.data.frame(pos)
dataset = cbind(dataset,pos)
pn = as.data.frame(table(dataset$pos))
colnames(pn) = c("class", "freq")
pie = ggplot(pn, aes(x = "", y=freq, fill = factor(class))) +
geom_bar(width = 1, stat = "identity") +
theme(axis.line = element_blank(),
plot.title = element_text(hjust=0.5)) +
labs(fill="class", x=NULL, y=NULL)
pie + coord_polar(theta = "y", start=0)
dataset$pos=NULL

#regressione logistica
#dtm
dtm = DocumentTermMatrix(corpus)
#removing sparse terms
sparse = removeSparseTerms(dtm, 0.98)
frq.words = as.data.frame(as.matrix(sparse))
colnames(frq.words) = make.names(colnames(frq.words))
dataset$Positive = as.factor(dataset$afinn>0)
frq.words$Positive = dataset$Positive
positive = gsub("TRUE","1",frq.words$Positive)
positive = as.data.frame(positive)
positive = gsub("FALSE","0",positive$positive)
positive = as.data.frame(positive)
frq.words = cbind(frq.words,positive)
frq.words$Positive=NULL
#splitting frq.words in training and testing set
library(caTools)
library(SnowballC)
set.seed(120)
split = sample.split(frq.words$positive,SplitRatio=0.7)
train = subset(frq.words,split==T)
test = subset(frq.words,split==F)
#logistic regression model
logReg = glm(positive~.,data=train,family=binomial)
summary(logReg)
Call:
glm(formula = positive ~ ., family = binomial, data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.0913   0.0001   0.0662   0.3192   2.5305
```

```
255
256 Coefficients:
257              Estimate Std. Error z value Pr(>|z|)
258 (Intercept) -0.66241    0.13370  -4.954 7.26e-07 ***
259 abl          0.27424    0.54721   0.501 0.616262
260 alexa        0.06875    0.25093   0.274 0.784089
261 also        -0.41696    0.52397  -0.796 0.426162
262 amaz         5.47542    1.42238   3.849 0.000118 ***
263 amazon      -0.15823    0.29566  -0.535 0.592534
264 answer      -1.18860    0.55147  -2.155 0.031137 *
265 anyth       -0.66486    0.63005  -1.055 0.291311
266 app         -0.06531    0.47677  -0.137 0.891049
267 ask          0.04270    0.35852   0.119 0.905206
268 awesom      19.73012  771.89579   0.026 0.979608
269 best        18.06874  789.87622   0.023 0.981750
270 better       2.94099    0.46118   6.377 1.81e-10 ***
271 bought      -1.41974    0.46755  -3.037 0.002393 **
272 buy         -0.25456    0.61766  -0.412 0.680237
273 call        -0.31681    0.46992  -0.674 0.500193
274 can         -0.11757    0.28370  -0.414 0.678578
275 cant        -0.88902    0.56435  -1.575 0.115185
276 christma     1.44067    0.70570   2.041 0.041203 *
277 command     -1.65499    0.57213  -2.893 0.003820 **
278 connect     -0.88438    0.37999  -2.327 0.019944 *
279 control     -0.49017    0.73643  -0.666 0.505670
280 day         -0.34965    0.46719  -0.748 0.454216
281 devic        0.02924    0.33267   0.088 0.929956
282 doesnt      -0.50716    0.41152  -1.232 0.217798
283 dont        -0.90538    0.47681  -1.899 0.057589 .
284 dot         -1.68953    0.40492  -4.172 3.01e-05 ***
285 easi         2.61799    0.51525   5.081 3.75e-07 ***
286 echo         0.03830    0.19243   0.199 0.842255
287 enjoy        1.94981    0.42510   4.587 4.50e-06 ***
288 even        -0.50122    0.44324  -1.131 0.258135
289 everi        0.36088    0.53437   0.675 0.499465
290 everyth      0.73419    0.46137   1.591 0.111533
291 expect      -0.42156    0.42121  -1.001 0.316913
292 famili       0.03988    0.63525   0.063 0.949938
293 far          0.47642    0.57903   0.823 0.410629
294 featur       0.35952    0.79091   0.455 0.649420
295 find         0.15911    0.51318   0.310 0.756522
296 first       -0.86512    0.51013  -1.696 0.089911 .
297 fun         18.88082  563.11573   0.034 0.973253
298 gen         -1.46428    0.30205  -4.848 1.25e-06 ***
```

```
299 generat      1.01412    0.69759    1.454  0.146014
300 get          0.27172    0.27304    0.995  0.319654
301 gift         2.30492    0.65377    3.526  0.000423 ***
302 give         0.66058    0.56719    1.165  0.244161
303 good         2.82269    0.42871    6.584  4.58e-11 ***
304 got         -0.97271    0.62969   -1.545  0.122409
305 great        4.28224    0.42541   10.066   < 2e-16 ***
306 happi        5.92139    1.65953    3.568  0.000360 ***
307 hear        -0.22972    0.49685   -0.462  0.643832
308 help         2.53756    0.56480    4.493  7.03e-06 ***
309 home        -0.53202    0.46958   -1.133  0.257226
310 hous        -1.09141    0.47483   -2.299  0.021530 *
311 improv       1.09713    0.57885    1.895  0.058044 .
312 inform       0.15805    0.60336    0.262  0.793356
313 just        -0.83229    0.33497   -2.485  0.012967 *
314 keep         1.58701    0.75944    2.090  0.036644 *
315 know        -0.26071    0.45032   -0.579  0.562621
316 learn        0.42212    0.54879    0.769  0.441785
317 light        1.75032    0.66518    2.631  0.008504 **
318 like         2.11100    0.28248    7.473  7.83e-14 ***
319 listen       0.31621    0.47984    0.659  0.509908
320 littl        0.53824    0.88430    0.609  0.542752
321 look        -0.28381    0.48567   -0.584  0.558982
322 lot          0.30353    0.55404    0.548  0.583793
323 love         3.61411    0.25367   14.247   < 2e-16 ***
324 make        -0.07050    0.54443   -0.129  0.896971
325 mani         1.11218    0.62539    1.778  0.075340 .
326 much         0.26234    0.36196    0.725  0.468593
327 music        0.07309    0.22434    0.326  0.744566
328 need        -0.36997    0.44582   -0.830  0.406617
329 new          0.21085    0.47169    0.447  0.654865
330 nice         4.21344    1.03597    4.067  4.76e-05 ***
331 now          0.01245    0.46161    0.027  0.978489
332 one         -0.37674    0.28072   -1.342  0.179580
333 order       -1.10422    0.55046   -2.006  0.044857 *
334 phone       -1.20369    0.59386   -2.027  0.042674 *
335 play         0.05718    0.32326    0.177  0.859611
336 product      0.17501    0.36849    0.475  0.634830
337 purchas      0.56454    0.49010    1.152  0.249362
338 qualiti      0.03210    0.42412    0.076  0.939677
339 question     1.13255    0.53983    2.098  0.035906 *
340 realli       1.60832    0.48951    3.286  0.001018 **
341 receiv       2.76708    1.00591    2.751  0.005944 **
342 recommend    2.49350    1.25072    1.994  0.046190 *
```

```
respons        0.20277    0.80816    0.251  0.801889
room           1.26327    0.62251    2.029  0.042427 *
say            0.82048    0.52170    1.573  0.115790
set           -0.60637    0.42484   -1.427  0.153498
smart          1.79513    0.55408    3.240  0.001196 **
song           0.54442    0.48631    1.119  0.262929
sound          0.63373    0.23377    2.711  0.006709 **
speaker        0.69073    0.52855    1.307  0.191266
still          0.20271    0.49107    0.413  0.679754
tell          -1.11830    0.46563   -2.402  0.016319 *
thing         -0.66660    0.33931   -1.965  0.049461 *
think         -0.31708    0.70062   -0.453  0.650856
time          -1.12769    0.30737   -3.669  0.000244 ***
tri           -0.61719    0.42703   -1.445  0.148374
turn          -1.12381    0.58044   -1.936  0.052849 .
understand    -0.32335    0.51844   -0.624  0.532822
updat         -2.01245    0.58100   -3.464  0.000533 ***
use            0.33449    0.23087    1.449  0.147392
voic           1.86487    0.68216    2.734  0.006261 **
volum          0.36887    0.62967    0.586  0.558004
want           1.17188    0.45668    2.566  0.010286 *
way           -0.99907    0.58920   -1.696  0.089951 .
weather        0.06571    0.54065    0.122  0.903261
well          -0.15419    0.41517   -0.371  0.710340
will           0.78082    0.47887    1.631  0.102990
wish           1.77790    0.76625    2.320  0.020326 *
without       -0.33923    0.68177   -0.498  0.618784
work           0.09208    0.24698    0.373  0.709281
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2148.8  on 2512  degrees of freedom
Residual deviance: 1023.3  on 2400  degrees of freedom
AIC: 1249.3

Number of Fisher Scoring iterations: 18

#prediction on testing set
predictTest = predict(logReg,type="response",newdata=test)
#mean probability
tapply(predictTest,test$positive,mean)
        0         1
```

```
0.4733229 0.9046324
#confusion matrix
table(test$positive,predictTest>0.5)
      FALSE TRUE
  0      97    67
  1      73   840
specificity=97/(97+67)
specificity
[1] 0.5914634
sensitivity=840/(840+73)
sensitivity
[1] 0.9200438
accuracy=(97+840)/(97+840+67+73)
accuracy
[1] 0.8700093
library(ROCR)
ROCpred = prediction(predictTest,test$positive)
ROCperf = performance(ROCpred,"tpr","fpr")
plot(ROCperf,colorize=T,print.cutoffs.at=seq(0.1,by=0.1),text.adj=c
    (-0.2,1.7))
auc = performance(ROCpred, measure = "auc")
auc = auc@y.values[[1]]
auc
[1] 0.8887345
#baseline
table(test$positive)
  0   1
164 913

#testing overfitting
predictTrain = predict(logReg,type="response",newdata=train)
tapply(predictTrain,train$positive,mean)
        0         1
0.4090150 0.9262275
table(train$positive,predictTrain>0.5)

      FALSE TRUE
  0    271   113
  1    106  2023
(2023+271)/(271+113+106+2023)
[1] 0.9128532

#CART
library(rpart)
```

```
library(rpart.plot)
frq.words$Positive = dataset$Positive
frq.words$positive=NULL
#splitting frq.words in training and testing set
library(caTools)
library(SnowballC)
set.seed(120)
split = sample.split(frq.words$Positive, SplitRatio=0.7)
trainset = subset(frq.words, split==T)
testset = subset(frq.words, split==F)
CART = rpart(Positive ~ ., data=trainset, method="class")
#plotting
prp(CART)
#prediction on testing set
predictCART = predict(CART, newdata=testset, type="class")
#confusion matrix
table(testset$Positive, predictCART)
       predictCART
        FALSE TRUE
  FALSE   109   55
  TRUE     68  845
overall.accuracy=(109+845)/(109+55+68+845)
overall.accuracy
[1] 0.8857939
#baseline
table(testset$Positive)
FALSE  TRUE
  164   913
> 913/(164+913)
[1] 0.8477252

#Random Forest
library(randomForest)
set.seed (120)
#random forest con 200 alberi
randomForest = randomForest(Positive ~ ., data=trainset, ntree=200)
#predizione sul testing set
predictRF = predict(randomForest, newdata=testset)
#matrice di confusione
table(testset$Positive, predictRF)
       predictRF
        FALSE TRUE
  FALSE    87   77
  TRUE     48  865
```

```
474 overall.accuracy=(87+865)/(87+77+48+865)
475 overall.accuracy
476 [1] 0.8839369
477
478 #k-fold cross-validation
479 library(caret)
480 library(e1071)
481 #k=10
482 numFolds = trainControl( method = "cv", number = 10 )
483 #range di valore per il cp
484 cpGrid = expand.grid( .cp = seq(0.01,0.5,0.01))
485 #validazione
486 train(Positive ~ ., data = trainset, method = "rpart",
487 +trControl = numFolds, tuneGrid = cpGrid )
488 CART2 = rpart(Positive ~ ., data = trainset, method="class", cp = 0.01)
489 plotcp(CART2)
490 predictCART2 = predict(CART2, newdata = testset, type = "class")
491 table(testset$Positive, predictCART2)
492         predictCART2
493          FALSE TRUE
494   FALSE    109    55
495   TRUE      68   845
496 accuracy=(109+845)/(109+845+55+68)
497 accuracy
498 [1] 0.8857939
```

Bibliografia

[1] Abiteboul, S. et al. (2000), "Data on the Web: from relations to semistructured data and XML". Morgan Kaufmann.

[2] Altman, D. G. e Bland, J. M. (1994), "Diagnostic tests. 1: Sensitivity and specificity" in *BMJ: British Medical Journal*, 308(6943), 1552.

[3] Amazon (2018), "ISBN e ASIN". Accessibile da: *https://www.amazon.it/gp/help/customer/display.html?nodeId=201889580*.

[4] Bai, A. (2018), "Chatbot e assistenti virtuali: nel 2020 svolgeranno il 25% delle operazioni di supporto alla clientela". Accessibile da: *https://pro.hwupgrade.it/news/mercato/chatbot-e-assistenti-virtuali-nel-2020-svolgeranno-il-25-delle-operazioni-di-supporto-alla-clientela_74355.html*.

[5] Bellman, R. (1978), "An introduction to artificial intelligence: can computer think?", (No. 04; Q335, B4).

[6] Bengio, Y. e Grandvalet, Y. (2004), "No unbiased estimator of the variance of k-fold cross-validation" in *Journal of machine learning research*, 5(Sep), 1089-1105.

[7] Bird, S. et al. (2009), "Natural language processing with Python: analyzing text with the natural language toolkit", *O'Reilly Media, Inc.*.

[8] lei, D.M. et al. (2003), "Latent dirichlet allocation" in *Journal of machine Learning research*, 3(Jan), 993-1022.

[9] Boiy, E. e Moens, M. F. (2009), "A machine learning approach to sentiment analysis in multilingual Web texts" in *Information retrieval*, 12(5), 526-558.

[10] Boldrini, N. (2018), "Cos'è l'Intelligenza Artificiale, perchè tutti ne parlano e quali sono gli ambiti applicativi". Accessibile da *www.ai4business.it/intelligenza-artificiale/intelligenza-artificiale-cose/*.

[11] Bradley, A. P. (1997), "The use of the area under the ROC curve in the evaluation of machine learning algorithms" in *Pattern recognition*, 30(7), 1145-1159.

[12] Breiman, L. (1996), "Bagging predictors" in *Machine learning*, 24(2), 123-140.

[13] Cambridge University (2009), "Stemming and lemmatization". Accessibile da *https://nlp.stanford.edu/IR-book/html/htmledition/stemming-and-lemmatization-1.html*.

[14] Canalys (2018),"Google beats Amazon to first place in smart speaker market". Accessibile da: *https://www.canalys.com/newsroom/google-beats-amazon-to-first-place-in-smart-speaker-market*.

[15] CCM (2018), "Linguaggi informatici - API". Accessibile da: *https://it.ccm.net/contents/185-linguaggi-informatici-api*.

[16] Celi (2018), "Natural Language Processing, Software e risorse per l'elaborazione linguistica". Accessibile da: *https://www.celi.it/tecnologia/natural-language-processing/*.

[17] DataFlair (2018), "Text Mining in Data Mining-Concepts, Process & Applications". Accessibile da: *https://data-flair.training/blogs/text-mining/*.

[18] Data skills (2018), "Tecniche di clustering". Accessibile da: *https://www.dataskills.it/tecniche-di-clustering/*

[19] Expert System (2017), "Text mining and analytics: how does it work?". Accessibile da: *http://www.expertsystem.com/text-mining-analytics-work/*.

[20] Edureka (2018), "Growth of Unstructured Data – Learn Hadoop". Accessibile da: *https://www.edureka.co/blog/10-reasons-to-learn-hadoop/growth-of-unstructured-data-learn-hadoop-edureka/*.

[21] Elmasri, R. e Navathe, S. (2010), "Fundamentals of database systems". Addison-Wesley Publishing Company.

[22] Esposito, F. et al. (1997), "A comparative analysis of methods for pruning decision trees" in *IEEE transactions on pattern analysis and machine intelligence*, 19(5), 476-491.

[23] Fauske, K. M. (2006), "Neural network". Accessibile da: *http://www.texample.net/tikz/examples/neural-network/*.

[24] Feldman, R. e Sanger, J. (2007), "The text mining handbook: advanced approaches in analyzing unstructured data". Cambridge university press.

[25] Gardner, H. E. (2000), "Intelligence reframed: Multiple intelligences for the 21st century". Hachette UK.

[26] Gartner (2017), "VPAs in the Enterprise". Accessibile da: https://www.gartner.com/newsroom/id/3790964.

[27] Goh, A. T. C. (1995), "Back-propagation neural networks for modeling complex systems" in *Artificial Intelligence in Engineering*, 9(3), 143-151.

[28] Goldberg, Y. (2017), "Neural network methods for natural language processing" in *Synthesis Lectures on Human Language Technologies*, 10(1), 1-309.

[29] Goldberg, D. E. eHolland, J. H. (1988), "Genetic algorithms and machine learning" in *Machine learning*, 3(2), 95-99.

[30] Govoni, L. (2018), "Rete Neurale, Deep Learning e principali applicazioni". Accessibile da: *https://lorenzogovoni.com/deep-learning-e-applicazioni/*.

[31] Harrell, F. E. (2001), "Ordinal logistic regression" in *Regression modeling strategies* (pp. 331-343). Springer, New York, NY.

[32] Hartigan, J. A. e Wong, M. A. (1979), "Algorithm AS 136: A k-means clustering algorithm" in *Journal of the Royal Statistical Society*, Series C (Applied Statistics), 28(1), 100-108.

[33] Hauke, J. e Kossowski, T. (2011), "Comparison of values of Pearson's and Spearman's correlation coefficients on the same sets of data" in *Quaestiones geographicae*, 30(2), 87-93.

[34] Hay, A. M. (1988), "The derivation of global estimates from a confusion matrix" in *International Journal of Remote Sensing*, 9(8), 1395-1398.

[35] Hebb, D. O. (1955), "Drives and the CNS (conceptual nervous system)" in *Psychological review*, 62(4), 243.

[36] Hennig, T. et al. (1997), "The impact of customer satisfaction and relationship quality on customer retention: A critical reassessment and model development" in *Psychology & marketing*, 14(8), 737-764.

[37] Hong, L. e Davison, B. D. (2010), "Empirical study of topic modeling in twitter" in *Proceedings of the first workshop on social media analytics* (pp. 80-88). ACM.

[38] Iezzi, F. (2018), "Analisi statistica di dati testuali". Accessibile da: *http://didattica.uniroma2.it/assets/uploads/corsi/39157/Analisi_di_dati_testuali.pdf*.

[39] ain, A.K. et al. (1999), "Data clustering: a review" in *ACM computing surveys* (CSUR), 31(3), 264-323.

[40] ain, A.K. et al. (1988), "Algorithms for clustering data".

[41] Knight, K. e Marcu, D. (2002), "Summarization beyond sentence extraction: A probabilistic approach to sentence compression" in *Artificial Intelligence*, 139(1), 91-107.

[42] Kohavi, R. (1995), "A study of cross-validation and bootstrap for accuracy estimation and model selection" in *Ijcai* (Vol. 14, No. 2, pp. 1137-1145).

[43] Koslowsky, S. (2010), "Database: Text Mining in Marketing". Accessibile da: *https://www.targetmarketingmag.com/article/text-mining-database-marketing-customer-service-analysis/all/*.

[44] Kuhn, M. e Johnson, K. (2013), "Applied predictive modeling" (Vol. 26). New York: Springer.

[45] Laboratorio statitica (2018), "Il coefficiente di correlazione di Spearman per ranghi". Accessibile da: *https:laboratoriostatistica.files.wordpress.com/2014/09/spearman.pdf*.

[46] Linguamatics (2018), "NLP text mining–the ten thousand foot view". Accessibile da: *https://www.linguamatics.com/what-is-text-mining-nlp-machine-learning*.

[47] Lison, P. (2015). "An introduction to machine learning".

[48] Liu, B. e Zhang, L. (2012), "A survey of opinion mining and sentiment analysis" in *Mining text data* (pp. 415-463). Springer US.

[49] Liu, B. (2012), "Sentiment analysis and opinion mining" in *Synthesis lectures on human language technologies*, 5(1), 1-167.

[50] oh, W. Y. (2011), "Classification and regression trees" in *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1), 14-23.

[51] Lu, H. et al. (2018), "Brain intelligence: go beyond artificial intelligence" in *Mobile Networks and Applications*, 23(2), 368-375.

[52] Luger, G. F. e Stubblefield, W. A. (1993), "Artificial intelligence: its roots and scope" in *Artificial intelligence: structures and strategies for*, 1-34.

[53] MacQueen, J. (1967), "Some methods for classification and analysis of multivariate observations" *in Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (Vol. 1, No. 14, pp. 281-297)

[54] Mao, J. e Jain, A. K. (1996), "A self-organizing network for hyperellipsoidal clustering (HEC)" in *Ieee transactions on neural networks*, 7(1), 16-29.

[55] Mitchell, R. (2015), "Web scraping with Python: collecting data from the modern web" in *O'Reilly Media, Inc.*

[56] Mullin, M. D. e Sukthankar, R. (2000), "Complete Cross-Validation for Nearest Neighbor Classifiers" in *ICML* (pp. 639-646).

[57] Munzert S. et al. (2014), "Automated data collection with R: A practical guide to web scraping and text mining" in *John Wiley & Sons.*.

[58] Neeraj, K. (2017), "What is the difference between Machine Learning and Deep Learning". Accessibile da: *https://medium.com/Say2neeraj/what-is-the-difference-between-machine-learning-and-deep-learning-5795e4415be9*.

[59] Özdemir, C. e Bergler, S. (2015), "Clac-sentipipe: Semeval2015 subtasks 10 b, e, and task 11" in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)* (pp. 479-485).

[60] Oxford Reference (2018), "Text Mining". Accessibile da: *http://www.oxfordreference.com/view/10.1093/oi/authority.20110803103357444*.

[61] Pang, B. e Lillian, L. (2008), "Opinion mining and sentiment analysis" in *Foundations and Trends® in Information Retrieval* 2.1–2, 1-135.

[62] Pang, B. e Lee, L. (2004), "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts" in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics* (p. 271), Association for Computational Linguistics.

[63] Pencina, M. J. et al.(2008), "Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond" in *Statistics in medicine*, 27(2), 157-172.

[64] Piatetsky-Shapiro, G. (1996), "Advances in knowledge discovery and data mining" (Vol. 21). U. M. Fayyad, P. Smyth, & R. Uthurusamy (Eds.). Menlo Park: AAAI press.

[65] Porcu, V. (2016), "Introduzione al machine learning con R".

[66] Porter, M. F. (2001) "Snowball: A language for stemming algorithms".

[67] Provost, F. J.,et al. (1998), "The case against accuracy estimation for comparing induction algorithms" in *ICML* (Vol. 98, pp. 445-453).

[68] Rao, V. (2013), "Introduction to Classification & Regression Trees (CART)". Accessibile su: *https://www.datasciencecentral.com/profiles/blogs/introduction-to-classification-regression-trees-cart*.

[69] Rissland, E. L. (1987), "Artificial Intelligence: knowledge representation". Cognitive science: An introduction, second printing.

[70] Robert, C. (2014), "Machine learning, a probabilistic perspective".

[71] Rouse, M. (2018), "Virtual Assistant (AI assistant)". Accessibile da: *https://searchcrm.techtarget.com/definition/virtual-assistant*.

[72] Russell, S. J. e Norvig, P. (2016), "Artificial intelligence: a modern approach", Malaysia. Pearson Education Limited.

[73] SAS (2018), "L'evoluzione del machine learning". Accessibile da: *https://www.sas.com/it_it/insights/analytics/machine-learning.html*.

[74] SAS (2018), "SAS Sentiment Analysis". Accessibile da: *https://www.sas.com/it_it/software/sentiment-analysis.html*.

[75] Sayad S. (2017), "Logistice Regression". Accessibile da: *http://www.saedsayad.com/logistic_regression.htm*.

[76] Schalkoff, R. J. (1997), "Artificial neural networks" (Vol. 1). New York: McGraw-Hill.

[77] Schmidhuber, J. (2015). "Deep learning in neural networks: An overview" in *Neural networks*, 61, 85-117.

[78] Shao, Y. e Lunetta, R. S. (2012), "Comparison of support vector machine, neural network, and CART algorithms for the land-cover classification using limited training data points" in I*SPRS Journal of Photogrammetry and Remote Sensing*, 70, 78-87.

[79] Statsoft (2018), "Random Forest". Accessibile da: *http://www.statsoft.com/Textbook/Random-Forest*.

[80] Statstutor (2018), "Spearman's correlation". Accessibile da: *http://www.statstutor.ac.uk/resources/uploaded/spearmans.pdf*.

[81] Steinbach, M. et al. (2000), "A comparison of document clustering techniques" in *KDD workshop on text mining* (Vol. 400, No. 1, pp. 525-526).

[82] Steinberg, D. e Colla, P. (2009) "The top ten algorithms in data mining", 9, 179.

[83] Tetko, I. V. et al. (1995), "Neural network studies. 1. Comparison of overfitting and overtraining" in *Journal of chemical information and computer sciences*, 35(5), 826-833.

[84] Traspinar, A. (2015), "Text Classification and Sentiment Analysis". Accessibile da: *http://ataspinar.com/2015/11/16/text-classification-and-sentiment-analysis/*.

[85] Upwork (2018), "What is Web Scraping and How Can You Use It?". Accessibile da: *https://www.upwork.com/hiring/for-clients/web-scraping-tutorial/*.

[86] Varathan, K. D. et al. (2017), "Comparative opinion mining: a review" in *Journal of the Association for Information Science and Technology*, 68(4), 811-829.

[87] Vinodhini, G. e Chandrasekaran, R. M. (2012). "Sentiment analysis and opinion mining: a survey".*International Journal*, 2(6), 282-292.

Introduzione

L'analisi dei dati testuali rappresenta oggi, per le aziende, uno degli orizzonti più importanti, sia in termini di volume che di rilevanza delle informazioni ottenibili. Con la crescente disponibilità e popolarità di risorse ricche di opinioni come i social media e i siti di recensione, sorgono nuove opportunità e sfide poichè è possibile utilizzare proattivamente le tecnologie dell'informazione per cercare e comprendere le opinioni altrui. In un mercato che offre un ventaglio sempre più ampio di scelte, le aziende, quanto mai preoccupate per la *customer loyalty* e la propria *web reputation*, necessitano di sistemi automatizzati per accedere alle informazioni di proprio interesse in modo semplice, rapido ed intuitivo. Oggigiorno, i dati vengono generati in una varietà di formati e ad una velocità che non mostra segni di rallentamento. Identificare le informazioni che possono facilitare il processo decisionale tra migliaia di documenti, pagine web e *social media feed* rappresenta un processo complesso e dispendioso in termini di tempo. Per tale motivo, l'informazione in forma testuale viene spesso ignorata o utilizzata solo parzialmente per supportare i processi organizzativi. In quest'ottica, le tecniche di *text mining* ed *opinion mining* aiutano a "digerire" i dati di testo in modo più efficiente offrendo l'opportunità di scoprire modelli interessanti e la capacità di trasformare le informazioni testuali in conoscenze utili per il *reputation management* dell'azienda[1]. Questo lavoro di ricerca si propone di rispondere a tale necessità, fornendo metodi e approcci per la categorizzazione automatica dei documenti e il rilevamento del *sentiment* all'interno di un corpo di recensioni disponibile online.

La prima sezione offre una panoramica sulle techine di *text mining* e *data mining*, in grado di identificare fatti, relazioni ed affermazioni che altrimenti rimarrebbero sepolti sotto una massa di dati testuali. In particolare, tramite l'impiego di algoritmi di *Natural Language Processing* si mostra come decifrare le ambiguità del linguaggio naturale e convertire dati testuali non strutturati in metadati.

La seconda sezione volge uno sguardo verso la disciplina dell'*Artificial Intelligence* cercando di definire cosa consente ad un sistema, quale uno *smart speaker*, di mostrare attività intelligente. In particolare, la trattazione riguarda i due principali metodi di apprendimento: gli algoritmi di *machine learning* che "allenano" i sistemi di intelligenza artificiale e il *deep learning* che permette di emulare la mente dell'uomo tramite l'impiego di algoritmi e di reti neurali artificiali progettate *ad hoc*.

[1] Expert System (2017), "Text mining and analytics: how does it work?". Accessibile da: *http://www.expertsystem.com/text-mining-analytics-work/*.

La terza sezione, infine, propone metodi di analisi statistica applicati tramite il linguaggio R e volti a decifrare e comprendere le informazioni contenute nel corpo di recensioni Amazon dello *smart speaker* Amazon Echo. Nel dettaglio, viene offerta una guida alla programmazione standard che va dal *web scraping*, attraverso cui convertire HTML *tag* in metadati poi memorizzati ed analizzati in locale all'interno di un database fino all'approccio *Bag of Words*, in cui ogni recensione viene trasformata in un vettore impiegabile come dato di input (o output) per algoritmi di *machine learning*. Due tecniche di *unsupervised machine learning*, il *Topic Modeling* e gli algoritmi di *clustering*, hanno permesso di identificare distribuzioni insite nell'insieme dei dati a disposizione evidenziando, in modi differenti ma al contempo equipollenti, quattro topic discussi all'interno della raccolta: l'utilizzo domestico del prodotto, la capacità del *device* di riprodurre musica, la qualità del prodotto e la semplicità ed il piacere derivante dall'utilizzo del prodotto. L'elaborato di tesi mostra anche come calcolare la *polarity* della raccolta estraendo il *sentiment* dal testo tramite tre lessici che, per quanto forniscano risultati differenti in senso assoluto, mostrano comunque traiettorie relative simili: un *overall sentiment* assolutamente positivo. Infine, dopo aver quantificato e caratterizzato il contenuto sentimentale dell'unità di testo, sono stati costruiti tre *classifier* predittivi di testo in grado di etichettare correttamente una nuova recensione scritta per il prodotto come positiva o negativa. Per ognuno dei tre modelli (regressione logistica, CART e Random Forest) è stata valutata l'accuratezza e la capacità di generalizzazione ad un set di dati indipendenti. Da una valutazione del *trade-off* tra accuratezza e complessità del modello e dopo un confronto con l'abilità predittiva della *baseline*, si deriva che il CART mostra performance più elevate, poichè più degli altri minimizza il costo di errata classificazione. Per testare il rischio di *overfitting* del CART è stata infine applicata una *k*-fold cross validation i cui risultati mostrano che il modello spiega la devianza presente nei dati originari senza adattarvisi in modo eccessivo.

1 Tecniche di Text Mining

Una parte importante del nostro comportamento di raccolta delle informazioni è sempre stata quella di scoprire cosa pensano gli altri. Con l'avvento della digitalizzazione, l'aumento dei social network e l'incremento della connettività, per le aziende sorgono nuove opportunità ed esigenze legate al *reputation management* e al comprendere la percezione online del brand. In questo contesto, il rilevamento automatico del *sentiment*, oggetto di discussione in questa sezione, può rappre-

sentare un ottimo strumento di monitoraggio. L'improvvisa esplosione di attività nell'area dell'opinione pubblica si è verificata, in buona parte, come diretta risposta all'impulso di interesse nei nuovi sistemi e nelle capacità di questi ultimi di fornire un mezzo per trattare direttamente con le opinioni quali oggetto di prima classe. Il compito è tecnicamente impegnativo ma praticamente molto utile: da una parte le aziende vogliono comprendere le opinioni dei consumatori, dalI'altra parte i potenziali clienti vogliono conoscere le opinioni degli utenti esistenti prima di utilizzare un servizio o acquistare un prodotto. Tuttavia, monitorare i siti di opinione e distillare le informazioni in essi contenute rimane un compito formidabile a causa del volume di testo, difficilmente processabile e decifrabile. Ricerche[2] mostrano come oggi circa l'80% delle informazioni online si presenti in forma non strutturata e, poichè il *forecast* al 2020 prevede un aumento della mole di dati non strutturati sia in valore relativo (85-90% del totale) che assoluto (40000 *exabytes*), le aziende cercano di affrontare questo immenso archivio dati tramite il *data mining* e le nuove applicazioni di *text mining*, con l'obiettivo di creare valore aziendale.

Il *text mining* è una tecnica che impiega strumenti di *data mining* in grado di identificare fatti, relazioni ed affermazioni che altrimenti rimarrebbero sepolti sotto una massa di dati testuali[3]. Queste informazioni vengono estratte e trasformate in dati strutturati tramite l'impiego di algoritmi di *Natural Language Processing* (NLP) che consentono al computer di "leggere" ed analizzare le informazioni codificate in linguaggio naturale[4], riconoscendo concetti simili, anche se espressi in modi differenti. La *sentiment analysis*, in particolare, si occupa del trattamento computazionale dell'opinione, del *sentiment* e della soggettività del testo nei confronti di entità e loro *feature*, fornendo tecniche ed approcci che promettono di attivare sistemi di ricerca di informazione orientati all'opinione pubblica[5]. Il rilevamento automatico del *sentiment*, come vedremo nella sezione 3.3, da un un punto di vista computazionale segue il processo[6] descritto nella Figura 1.

[2]Edureka (2018), "Growth of Unstructured Data – Learn Hadoop". Accessibile da: *https://www.edureka.co/blog/10-reasons-to-learn-hadoop/growth-of-unstructured-data-learn-hadoop-edureka/*.

[3]Feldman, R. e Sanger, J. (2007), "The text mining handbook: advanced approaches in analyzing unstructured data". Cambridge university press.

[4]Celi (2018), "Natural Language Processing, Software e risorse per l'elaborazione linguistica". Accessibile da: *https://www.celi.it/tecnologia/natural-language-processing/*.

[5]Pang, B. e Lillian, L. (2008), "Opinion mining and sentiment analysis" in *Foundations and Trends® in Information Retrieval* 2.1–2, 1-135.

[6]DataFlair (2018), "Text Mining in Data Mining-Concepts, Process & Applications". Accessibile da: *https://data-flair.training/blogs/text-mining/*.

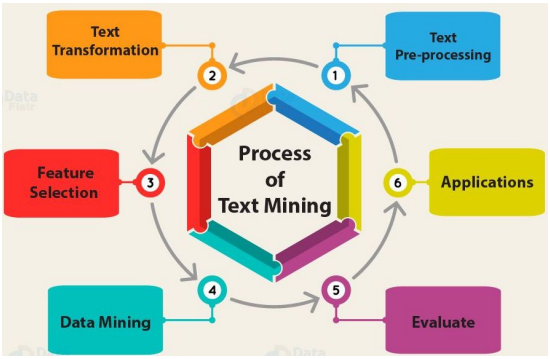

Figura 1. *Processo di text mining*. L'interpretazione del testo prevede una serie di fasi preliminari che riguardano l'estrazione, la codifica dei dati, la word normalization e l'applicazione degli strumenti di data mining.

2 Strumenti di Artificial Intelligence: i Virtual Assistant

Nel 1950 Alan Turing pubblicò un articolo dal titolo "*Computing Machinery and Intelligence*", in cui propose un criterio, oggi definito "Test di Turing", in grado di determinare se una macchina è in grado di pensare o meno. Per soddisfare questo criterio un software deve fingere di essere umano in una conversazione in tempo reale in modo che l'interlocutore non sia in grado di distinguere, basandosi solo sul contenuto della conversazione, se stia conversando con un programma o con un essere umano. Definire cosa sia esattamente l'l'*Artificial Intelligence* (AI) è un compito arduo. Nella sua accezione puramente informatica, l'AI rappresenta la disciplina che racchiude le teorie e le tecniche pratiche (programmazione e progettazione di sistemi hardware e software) per lo sviluppo di algoritmi che consentano alle macchine di mostrare attività intelligente. Il funzionamento tecnologico si sostanzia attraverso quattro differenti livelli: comprensione, ragionamento, apprendimento e *human machine interaction*[7]. Ciò che caratterizza l'AI, da un punto di vista tecnologico e metodologico, è proprio il metodo di apprendimento con cui l'intelligenza diventa abile. In particolare, i due modelli di apprendimento sono il *machine learning* e il *deep learning*. Gli algoritmi del primo permettono ad una macchina di apprendere senza essere stata preventivamente programmata[8]. In particolare, distinguiamo tre approcci: il *supervised machine learning*, dove le etichette sono create dall'addestratore per rendere la macchina capace di scoprire relazioni tra input ed etichette; il *unsupervised machine learning* dove le etichette non sono disponibili e si chiede alla macchina semplicemente di trovare dei cluster all'interno dei dati e, infine, il *reinforcement learning* che sviluppa algoritmi in grado di apprendere

[7]Russell, S. J. e Norvig, P. (2016), "Artificial intelligence: a modern approach". Malaysia; Pearson Education Limited.

[8]Goldberg, D. E. e Holland, J. H. (1988). "Genetic algorithms and machine learning" in *Machine learning*, 3(2), 95-99.

ed adattarsi alle mutazioni dell'ambiente. Il secondo metodo di apprendimento, sottoinsieme del *machine learning* è il *deep learning*, dove l'apprendimento avviene tramite reti neurali artificiali[9] (*Deep Artificial Neural Networks*).

Tra le diverse applicazioni dell'AI figurano i Virtual Assistant (VA), agenti software in grado di comprendere i comandi vocali in linguaggio naturale ed eseguire attività richieste dall'utente. Il set di funzionalità comprende l'esecuzione di dettati, la lettura di messaggi di testo, la ricerca, la pianificazione, gestione di chiamate telefoniche, oltre al fornire tutte quelle informazioni che normalmente verrebbero ricercate in un browser. I VA sono in genere programmi basati sul *cloud* che richiedono il funzionamento di dispositivi e/o applicazioni connessi ad Internet: l'utente rivolge una richiesta che il VA elabora ed archivia nel *cloud* stesso. In particolare, per abbinare il testo dell'utente o l'input vocale ai comandi eseguibili e avviare un processo di apprendimento si utilizzano tecniche di NLP, piattaforme di riconoscimento e sintesi vocale, algoritmi di *machine learning* che, tutti insieme, si combinano per garantire la massima flessibilità ed adattabilità agli strumenti utilizzati dal consumatore. I dati parlano chiaro[10] attualmente 2 ricerche su 10 effettuate da mobile su Google sono vocali e si prevede che entro il 2020 il 50% delle ricerche avverranno senza l'utilizzo dello schermo. Spesso tali VA sono incorporati all'interno di *device*, definiti *smart speaker* che, attivati per mezzo di una "*hot-word*", sono in grado di offrire azioni interattive. Secondo una ricerca di Canalys[11] sono l'accessorio di elettronica di consumo più in crescita con una previsione di vendite di circa 56,3 milioni di unità nel 2018. Sono stati poco meno di 35 milioni nel 2017, furono poco più di 5 milioni nel 2016.

3 Metodologia

La classificazione automatica dei documenti testuali per mezzo di algoritmi di *machine learning* conta un grande numero di applicazioni, fornendo soluzioni ai problemi delle aziende che vogliono accedere alle informazioni di proprio interesse in modo semplice, rapido ed intuitivo. L'esigenza deriva dalla crescita esplosiva del web e dal fermento creatosi attorno ai social media, ai siti di recensione e ad ogni

[9] Schmidhuber, J. (2015). "Deep learning in neural networks: An overview" in *Neural networks*, 61, 85-117.

[10] Gartner (2017), "Gartner Says Worldwide Spending on VPA-Enabled Wireless Speakers Will Top $3.5 Billion by 2021". Accessibile da: *https://www.gartner.com/newsroom/id/3790964*.

[11] Canalys (2018),"Google beats Amazon to first place in smart speaker market". Accessibile da: *https://www.canalys.com/newsroom/google-beats-amazon-to-first-place-in-smart-speaker-market*.

strumento portatore della voce del consumatore. Per tale ragione le aziende, sempre più preoccupate del processo di fidelizzazione del *prospect*, confidano oggi nell'analisi testuale automatizzata volta a monitorare la *web reputation* del brand. I principali *benefit* portati dall'analisi testuale sono: "l'abilità di monitorare i sentimenti e le opinioni in tempo reale, una chance per ottenere feedback e nuove idee, ottenere aggiornamenti sulla posizione dell'azienda e su quella dei suoi *competitor* all'interno dell'ambiente competitivo, l'abilità di quantificare le percezioni del mercato[12]".

3.1 Data Scraping e BoW

La raccolta automatizzata di dati da Internet è vecchia quasi quanto Internet stesso. Il processo, detto *web scraping*, ci consente di estrarre i dati presenti su un sito web in forma non strutturata (HTML *tag*) e convertirli in metadati, memorizzati ed analizzati in locale all'interno di un database[13]. Nel caso in analisi, per operare lo *scraping* dei dati dall'HTML della pagina web Amazon del prodotto Echo, sono state applicate funzioni del pacchetto rvest, comunemente utilizzato per la manipolazione di codici HTML e XML. Tutto il lavoro è stato svolto da un programma automatico (pezzo di codice), detto *scraper*, che opera seguendo quattro step:

1. Invia una *get query* ad uno specifico sito web.
2. Richiede i dati, di solito nella forma di HTML.
3. Analizza il documento HTML per estrarre le informazioni necessarie.
4. Converte i dati in un qualunque formato.

Il *data frame* ottenuto, nominato "dataset", è stato ripulito delle variabili prive di contenuto informativo ai fini della ricerca. Al termine del *data cleaning*, il dataset conta 3590 osservazioni (recensioni) per 2 categorie informative (vettori stars e comments). Come trasformare ora il testo in numeri? Il problema sta nel fatto che gli algoritmi di *machine learning*, per lavorare, preferiscono input ed output ben definiti a lunghezza fissa. In questo contesto si inserisce il *Bag of Words* (BoW), modello di analisi e rappresentazione testuale in grado di estrarre date *feature* da documenti testuali ed inserirle prima in un *corpus*, poi all'interno di una matrice a due dimensioni. Con la creazione della matrice BoW, ogni recensione viene convertita in un vettore numerico impiegabile come dato di input (o output) per

[12]SAS (2018), "SAS Sentiment Analysis". Accessibile da: *https://www.sas.com/it_it/software/sentiment-analysis.html*.

[13]Mitchell, R. (2015), "Web scraping with Python: collecting data from the modern web" in *O'Reilly Media, Inc.*

gli algoritmi di *machine learning*. In particolare, il processo di conversione prevede prima il pre-processamento del *text corpus* (fine alla riduzione del dizionario[14] con rimozione della punteggiatura,dei caratteri numerici, delle *stopword* e *stemming*), poi la creazione di una Term Document Matrix ("tdm"), in cui le colonne corrispondono alla lista delle recensioni, ogni riga rappresenta uno dei termini contenuti nelle recensioni ed ogni cella (valore della matrice) contiene il numero di apparizioni di quel dato termine in quel dato documento. Si noti come, seguendo tale logica, si perde ogni informazione sull'ordine, struttura del documento e relazione semantica tra le parole. Ma ciò non importa perchè è unicamente la frequenza di occorrenze delle parole ad essere utilizzata come *feature* per allenare i *text classifier*.

3.2 Topic Modeling Method e Algoritmi di Clustering

Nell'era dell'informazione che stiamo vivendo l'ammontare di materiale interattivo testuale (non strutturato) con cui ci interfacciamo ogni giorno risulta semplicemente aldilà delle nostre capacità di processamento. Per questo motivo, è necessario adottare metodi di organizzazione e classificazione testuale che ci aiutino a decifrare e comprendere le informazioni provenienti dal mercato. In questa sezione tratteremo due tecniche di *unsupervised machine learning* che permettono di creare gruppi di *item* (recensioni) in base ad un dato criterio di classificazione: il Topic Modeling Method e gli algoritmi di clustering.

Il Topic Modeling, tramite l'approccio Latent Dirichlet Allocation (LDA) consente di scoprire topic latenti all'interno di una collezione di documenti, associare questi ultimi ai topic individuati ed infine utilizzare l'associazione calcolata per organizzare ed interpretare i dati testuali[15]. In particolare, l'algoritmo determina, a partire da una DocumentTermMatrix (trasposta della tdm vista nella sezione 3.1), sia la probabilità di appartenenza *per-topic-per-word* β, che la probabilità *per-document-per-topic* γ restituendo i topic latenti nella raccolta di recensioni in forma di distribuzioni multinominali con probabilità di appartenenza associate.

Come mostrato nella Figura 2, le parole più comuni all'interno del topic 1 suggeriscono un orientamento verso l'utilizzo domestico del prodotto; i termini più comuni all'interno del topic 2 suggeriscono un'orientamento verso la capacità del *device* di riprodurre musica; le parole più comuni all'interno del topic 3 suggeriscono

[14]Traspinar, A. (2015), "Text Classification and Sentiment Analysis". Accessibile da: *http://ataspinar.com/2015/11/16/text-classification-and-sentiment-analysis/*.

[15]Hong, L. e Davison, B. D. (2010), "Empirical study of topic modeling in twitter" in *Proceedings of the first workshop on social media analytics* (pp. 80-88), ACM.

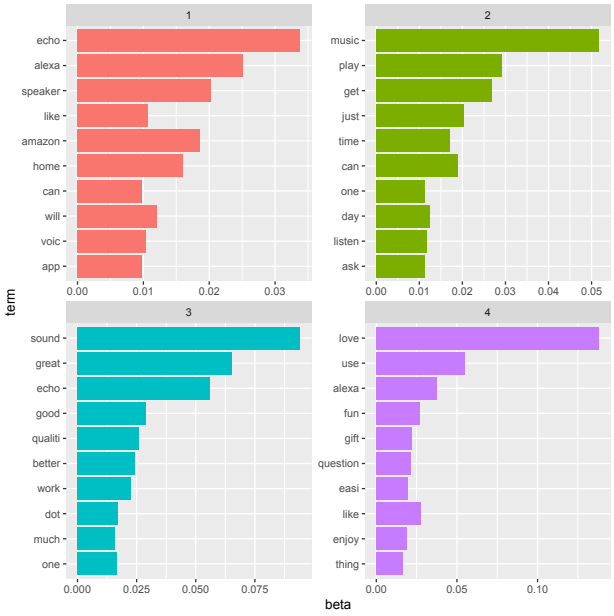

Figura 2. *4-topic LDA model*. Distribuzioni multinominali composte dai 10 termini più frequenti all'interno di ognuno dei quattro topic.

un orientamento verso la qualità del prodotto, il suono e il suo funzionamento. Il quarto topic, invece, raccoglie termini che suggeriscono un orientamento verso la semplicità e il piacere derivante dall'utilizzo del prodotto. Ora che il metodo del Topic Modeling ha generato una rappresentazione delle recensioni in un *topic space*, l'idea è utilizzare tale rappresentazione come "base" per performare gli algoritmi di clustering in modo da verificare se l'associazione tra recensioni e topic sarà equivalente.

La *cluster analysis* è una tecnica di analisi multivariata che consente di classificare un insieme di *pattern* in sottoinsiemi minimizzando la "lontananza logica" interna ad ogni gruppo (devianza *within-groups*) e massimizzando quella tra i gruppi[16] (devianza *between-groups*). Per quantificare tale lontananza logica, nel nostro caso si è adottata come misura di similarità la frequenza di occorrenza delle parole riportata nella matrice tdm priva dei termini sparsi. In particolare è stato applicato sia l'algoritmo gerarchico agglomerativo che il *k*-medie. Il primo ha prodotto, tramite la funzione hclust(), un dendrogramma (v. Fig. 3), ovvero un grafo ad albero rappresentante il raggruppamento annidato delle unità statistiche[17]. Il metodo utilizzato per calcolare la distanza tra le unità è stato quello di Ward che, ad ogni passo, tende ad ottimizzare la partizione ottenuta tramite l'aggregazione di due

[16]Jain, A.K. et al. (1999), "Data clustering: a review" in *ACM computing surveys*(CSUR), 31(3), 264-323.

[17]Steinbach, M. et al. (2000), "A comparison of document clustering techniques" in *KDD workshop on text mining* (Vol. 400, No. 1, pp. 525-526).

elementi garantendo la minimizzazione della devianza *within-groups*. Ai fini della ricerca il grafo è stato tagliato in corrispondenza del terzo *grouping level*. L'algoritmo del *k*-means, invece, segue un processo iterativo di riallocazione dei *k* poli cercando, ad ogni partizione, di minimizzare la devianza *within-groups*[18]. La Figura 4 mostra i cluster generati dalla funzione kmeans() al convergere dell'algoritmo.

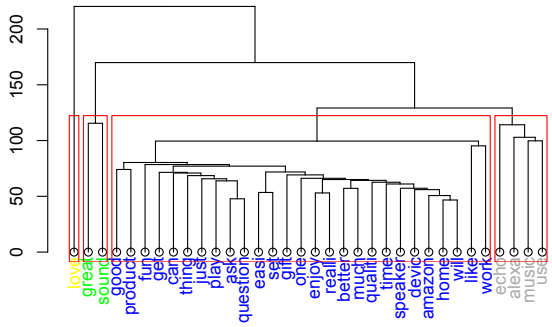

Figura 3. *Dendogramma*

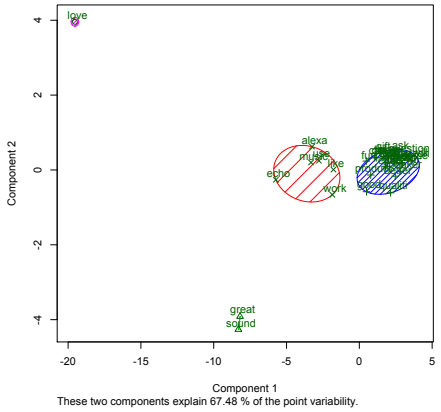

Figura 4. *Clusplot del k-means*

Ogni parola è etichettata con un colore che indica l'appartenenza ad uno dei quattro cluster. Ora è possibile fare una comparazione tra Topic Modeling e algoritmi di clustering, evidenziando similarità e differenze. Prima di tutto, una chiarificazione è necessaria: come sappiamo l'algoritmo di clustering è stato applicato alla matrice costituita solo dai *not sparse term*; nel Topic Modeling Method, invece, è stata utilizzata l'intera BoW. Questa rappresenta una falsa dissimilarità perchè il Topic Modeling, utilizzando l'approccio LDA, crea topic prendendo in considerazione solo i termini che appaiono nella raccolta con maggior frequenza. Per quanto riguarda le somiglianze, è possibile notare come ci sia una netta corrispondenza tra i gruppi di parole che costituiscono i topic nella Figura 2 e i cluster di parole mostrati nelle Figure 3 e 4.

3.3 Sentiment Polarity

Come visto nella sezione 1, la gestione e l'interpretazione dei contenuti condivisi sul web fornisce importanti implicazioni per le aziende: analizzando e classificando le opinioni, è possibile monitorare l'atteggiamento mutevole del pubblico. Pertanto,

[18]Hartigan, J. A. e Wong, M. A. (1979), "Algorithm AS 136: A k-means clustering algorithm" in *Journal of the Royal Statistical Society*, Series C (Applied Statistics), 28(1), 100-108.

l'analisi del *sentiment* implica l'identificazione di tre elementi tra loro correlati[19], argomento di trattazione in quest sezione: espressioni di *sentiment*, polarità e forza delle espressioni, relazione tra *sentiment* e prodotto. Il primo passo è estrarre, tramite gli strumenti offerti dal pacchetto tidytext, tutte le *sentword* più rilevanti contenute nel corpo delle recensioni (v. Fig. 5), in modo da disporre di una prima misura indicativa sulla polarità della raccolta.

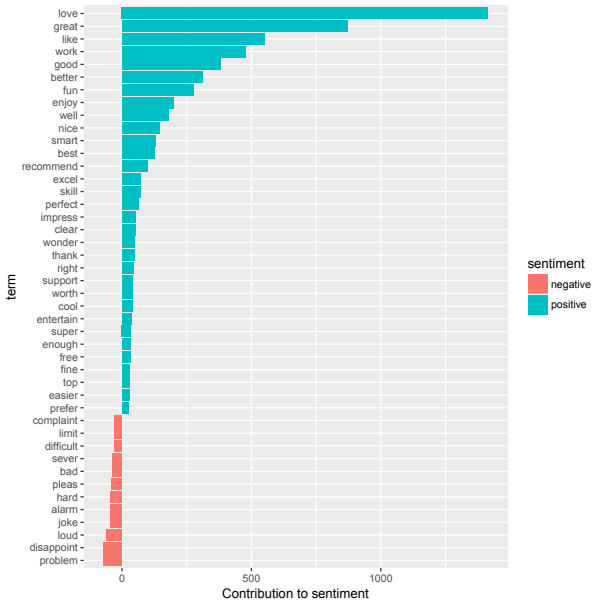

Figura 5. *Termini con la maggior contribuzione al sentiment*. L'impatto è misurato considerando la frequenza di occorrenza del termine: più si ripete, maggiore sarà il suo contributo al sentiment generale.

Il passo successivo è identificare, quantificare e caratterizzare il contenuto sentimentale dell'unità di testo confrontando i risultati ottenuti con tre differenti lessici (basati su unigrammi) disponibili nel pacchetto syuzhet: "afinn", "bing" e "nrc". Inserendo uno dei tre lessici come dato di input alla funzione get_sentiment(), ad ogni termine presente nelle recensioni viene assegnato un punteggio secondo i criteri dello specifico lessico. In particolare, il metodo afinn assegna alle parole un punteggio tra -5 (molto negativo) e 5 (molto positivo), il metodo bing categorizza i termini in positivi o negativi secondo una classificazione binaria (-1 se negativo, +1 se positivo), il metodo nrc, infine, non assegna punteggi alle parole di *sentiment* ma semplicemente le categorizza associandole ad una classe emotiva (v. Fig. 6).

Così come il lessico nrc riporta un prevalere di sentimenti positivi all'interno della raccolta, anche cumulando i punteggi assegnati alle parole che compongono ogni singola recensione secondo i metodi afinn e bing si ottiene un valore di

[19] Pang, B. e Lee, L. (2004), "A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts" in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics* (p. 271), Association for Computational Linguistics.

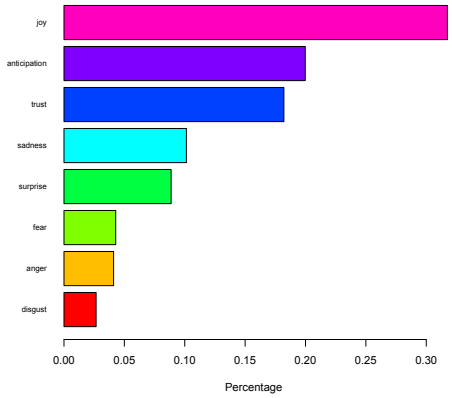

Figura 6. *Distribuzione delle emozioni*. Distribuzione delle otto emozioni fornite dal lessico nrc all'interno della raccolta di recensioni. L'asse delle ascisse indica la percentuale delle recensioni caratterizzate dalla specifica classe emotiva annoverata in ordinata.

overall sentiment assolutamente positivo e pari, rispettivamente, a 15044 e 5999. In conclusione, per quanto i tre lessici forniscano risultati differenti in senso assoluto, mostrano comunque la stessa traiettoria in termini di polarità.

3.4 Algoritmi di Supervised Machine Learning

Nell'ambito del *text mining*, applicare tecniche di *supervised machine learning* volte ad inferire una funzione o addestrare un classificatore su dati di *training* permette alle aziende di automatizzare il processo di estrazione delle informazioni da dati di testo. Ma cosa è un classificatore? Un *classifier* è una funzione che mappa un'istanza non etichettata in un'etichetta usando strutture di dati interni. In questa sezione vediamo come costruire modelli di regressione logistica, CART e Random Forest, in grado di classificare correttamente una nuova recensione scritta per il prodotto (assegnandole un'etichetta di appartenenza: positiva o negativa) valutandone accuratezza e capacità di generalizzazione. Per valutare l'abilità del classificatore probabilistico di generalizzare a *out-of-sample data*, tutti i modelli sono stati prima costruiti sul *training set* e poi testati eseguendo stime sul *testing set*[20]. I modelli, inoltre, assumono come variabile dipendente categorica un vettore che indica valore positivo se il punteggio afinn della specifica recensione è risultato positivo, valore negativo nel caso complementare.

Un primo *classifier* in grado di risolvere il problema di classificazione preannunciato è il modello di regressione logistica, utilizzato quando si è interessati a studiare o analizzare la relazione causale tra una variabile dipendente dicotomica (codificata

[20]Shao, Y. e Lunetta, R. S. (2012), "Comparison of support vector machine, neural network, and CART algorithms for the land-cover classification using limited training data points" in I*SPRS Journal of Photogrammetry and Remote Sensing*, 70, 78-87.

come 0-1) e una o più variabili indipendenti[21]. La funzione glm(), operante sul *training set* di una Document Term Matrix priva dei termini sparsi, accoglie in input come predittori tutti i vocaboli presenti nella matrice, rappresentati come vettori numerici. In questo senso è la frequenza di occorrenza dei termini corrispondenti alle recensioni etichettate come positive o negative (celle della matrice) a determinare l'influenza che le singole variabili indipendenti hanno sulla variabile risposta (quindi sul determinare se la recensione sarà positiva o meno). Poichè vogliamo una predizione binaria, per convertire le probabilità in una categoria è necessario un valore soglia (*cut-off*) rispetto al quale decidere se la recensione apparriene alla classe positiva o negativa. La matrice di confusione[22] generata da un livello soglia fissato a 0.5 ha restituito un valore di accuratezza del modello pari 0.8700093, ben più alto rispetto alla baseline (modello che fornisce come previsione sempre il valore più frequente) pari a 0.8477252. Per la scelta della soglia viene in aiuto la curva ROC[23] che come mostrato nella Figura 7, grafica i punti di coordinata $(fpr_i; tpr_i)$.

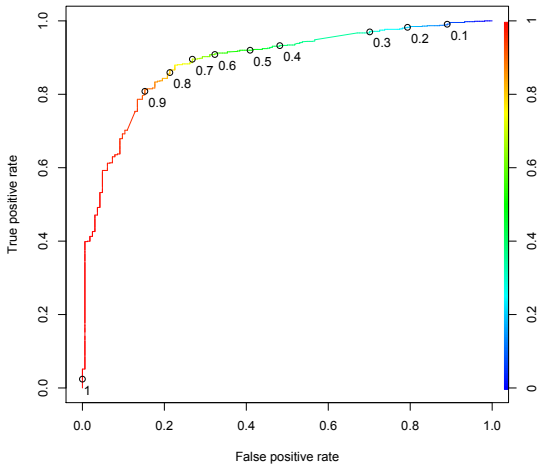

Figura 7. *Curva ROC*. Risultante di tutte le possibili scelte del cut-off mappato ad intervalli di 0.1. L'area sotto la curva, pari a 0.8887345, indica la probabilità di classificare correttamente una nuova recensione scelta randomicamente.

Gli alberi decisionali costituiscono il modo più semplice di classificare un set di osservazioni in un numero finito di classi. Questo è il motivo per cui, per predire se una recensione sarà positiva o no, ho applicato il CART, algoritmo che partiziona lo spazio dei dati in modo ricorsivo e binario (*split*) e, all'interno di ogni partizione,

[21] Harrell, F. E. (2001), "Ordinal logistic regression" in *Regression modeling strategies* (pp. 331-343). Springer, New York, NY.

[22] Provost, F. J.,et al. (1998), "The case against accuracy estimation for comparing induction algorithms" in *ICML* (Vol. 98, pp. 445-453).

[23] Pencina, M. J. et al. (2008), "Evaluating the added predictive ability of a new marker: from area under the ROC curve to reclassification and beyond" in *Statistics in medicine*, 27(2), 157-172.

adatta un semplice modello di predizione[24]. Il processo è iterato fino al raggiungimento dell'elemento foglia, etichettato con le differenti modalità che la variabile risposta può assumere[25]: TRUE se la recensione è positiva, FALSE in caso contrario. Utilizzando la stessa matrice impiegata per il classificatore logistico, l'output del modello sul *training set* è rappresentato nella Figura 8.

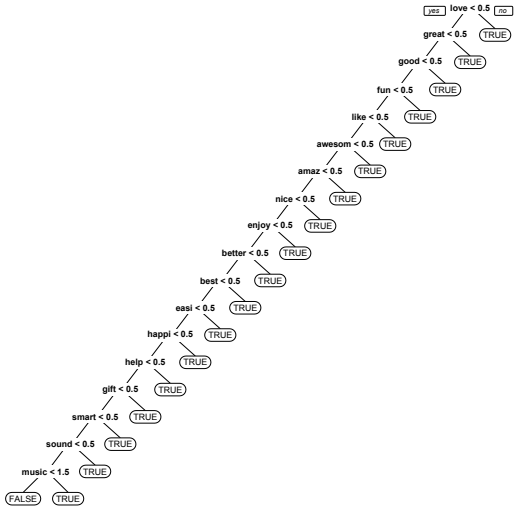

Figura 8. *CART*. Gerarchia di decisioni e conseguenze in base alla frequenza di occorrenza dei termini. Se il termine love è maggiore di .5, allora la recensione è classificata come positiva; in caso contrario, seguire lo *split* fino al termine "*great*". La logica predittiva è interata fino all'ultimo *split*.

Come leggere l'albero? Il senso è intuitivamente il seguente: se il termine "*love*" occorre in una recensione, è più probabile che la stessa sia positiva; se "*love*" non appare ma il termine "*great*" è presente, la recensione sarà probabilmente positiva. Il passo successivo è stato verificare la bontà delle predizioni fatte dal modello sul *testing set*. La matrice di confusione generata ha restituito un valore di accuratezza pari a 0.8857939, ben più alto rispetto alla *baseline* (0.8477252) e al classificatore logistico (0.8700093); di conseguenza, il CART fornisce prestazioni di classificazione più elevate. Spesso possono presentarsi alcuni problemi nella costruzione di alberi di classificazione binari e nella stima del costo di errata classificazione ad essi associato. Il rischio è legato alla possibilità che il CART si adatti eccessivamente ai dati di *training*. Per stimare quanto le previsioni fatte dal modello sono risultate accurate, la scelta più appropriata è la *k*-fold cross validation stratificata, tecnica che prevede diversi cicli di convalida incrociata utilizzando molteplici *training* e *testing set*[26]. Dividendo il set di dati in *k*=10 *fold*, l'output è stato un cp (*complexity parameter*)

[24]Loh, W. Y. (2011), "Classification and regression trees" in *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(1), 14-23.

[25]Breiman, L. (1996), "Bagging predictors" in *Machine learning*, 24(2), 123-140.

[26]Mullin, M. D. e Sukthankar, R. (2000), "Complete Cross-Validation for Nearest Neighbor

pari a 0.01, valore cui è associato il livello massimo di accuratezza. Tale valore, utilizzato per definire il livello di *pruning*, è stato utilizzato per costruire un nuovo modello CART. Verificando la predizione sul *testing set*, il livello di accuratezza del modello calcolato sulla matrice di confusione è pari a 0.8857939, quindi il CART originario non presenta alcun problema di *overfitting* sui dati.

Un classificatore in grado di migliorare l'accuratezza del modello CART è il Random Forest, metodo di *ensemble learning* che opera costruendo una moltitudine di alberi decisionali, ognuno dei quali sviluppato da un *bootstrap sample* estratto dai dati di *training*[27]. Il Random Forest, oltre a gestire le complicazioni causate dall'elevata dimensionalità dei dati, può gestire con successo anche i problemi derivanti da multicollinearità; questo perchè, a differenza del CART, risulta veloce ed insensibile ai problemi di *overfitting*. Nel caso pratico, utilizzando la funzione randomForest(), è possibile costruire il modello sul *training set* generando 200 differenti alberi (il numero di alberi definito è un *hyper-parameter*). La verifica della bontà delle predizioni sul *testing set* ha generato una matrice di confusione con accuratezza complessiva pari a 0.8839369, valore più alto rispetto alla *baseline* (0.8477252) ma non rispetto al CART (0.8857939). La scelta del modello da adottare richiede di ponderare il *trade-off* tra accuratezza e complessità del modello. Per tale ragione, in questo caso è preferibile il modello CART, poichè più semplice ed immediato da interpretare.

Conclusioni

Il lavoro di tesi ha seguito un approccio quantitativo nella guida all'analisi testuale ed ha esaminato tutti i risultati rispetto alle domande di ricerca, inerenti al fornire strumenti per la raccolta automatica di dati testuali che permettano alle aziende di monitorare in *real-time* la propria *web reputation* e la *brand perception*. La scelta di utilizzare come metodo di ricerca la raccolta automatizzata dei dati testuali tramite un linguaggio di programmazione consente di superare le limitazioni tipiche di un sondaggio, attività dispendiosa in termini di tempo ed accessibilità, con risultati non sempre accurati.

In questa ricerca è stato presentato un approccio all'analisi testuale a livello di topic che raggruppa le parole più frequenti in cluster associati a differenti topic discussi. I metodi di Topic Modeling e clustering hanno permesso, seguendo approcci

Classifiers" in *ICML* (pp. 639-646).

[27]Statsoft (2018), "Random Forest". Accessibile da: *http://www.statsoft.com/Textbook/Random-Forest*.

differenti, di comprendere e sintetizzare la raccolta testuale fornendo una finestra pubblica su quali sono gli aspetti del prodotto di maggior interesse per i consumatori: l'utilizzo domestico del prodotto, la capacità del *device* di riprodurre musica, la qualità del prodotto e la semplicità ed il piacere derivanti dall'utilizzo dello *smart speaker*. In particolare, poichè i topic utilizzati nel linguaggio naturale ammettono *overlapping* di parole, il Topic Modeling ha superato le limitazioni tipiche dell'*hard clustering*. Scoprire topic latenti e regole di associazione all'interno della collezione ha consentito di affrontare il problema della categorizzazione della polarità del *sentiment* e fornire *text classifier* predittivi. Dopo aver identificato, quantificato e caratterizzato il contenuto sentimentale dell'unità di testo (*overall polarity*), la conclusione raggiunta è che il *sentiment* che i recensori mostrano nei confronti del prodotto è decisamente positivo; questo potrebbe voler dire che le attese che i consumatori avevavo non sono state disattese dopo l'acquisto. Il confronto tra i risultati ottenuti con tre differenti lessici ci ha fornito una prospettiva a più ampio raggio per meglio comprendere l'orientamento emozionale delle recensioni. Infine è stato presentato un metodo per la collezione automatica di un *text corpus* utilizzabile per allenare *sentiment classifier*, in grado di identificare automaticamente la polarità per una nuova recensione raggiungendo risultati significativamente migliori rispetto alla *baseline*. Si è constatato che i classificatori di *sentiment* dipendono fortemente da domini o argomenti e nessuno dei modelli di classificazione supera in modo consistente l'altro. Si è comunque scoperto che il modello CART garantisce prestazioni più elevate rispetto al modello di regressione logistica e al Random Forest.

Nelle analisi fatte, abbiamo adottato come misura di similarità la frequenza di occorrenza dei termini. L'analisi delle relazioni basate sulla distanza ovviamente ha dei limiti. Ad esempio, anche quando un termine soggetto e una *sentiment word* sono contenuti nella stessa frase e posti vicini l'uno all'altro, il termine soggetto e il termine sentimento potrebbero non essere affatto correlati. La maggior parte delle applicazioni di *sentiment analysis* mira a classificare l'intero documento in positivo o negativo presupponendo che, ad essere oggetto di tutte le espressioni di sentimento sia il soggetto della raccola di recensioni. in realtà, espressioni che violano questa ipotesi confondono il giudizio di classificazione. Al contrario, analizzando le relazioni tra espressioni di sentimento e soggetti, è possibile fare analisi più approfondite e precise. In quest'ottica, il compito di uno studio successivo potrebbe essere analizzare frammenti di testo in modo da identificare relazioni semantiche tra soggetti ed espressioni di *sentiment* a loro relative; in questo caso, la polarità del *sentiment* potrebbe risultare completamente diversa a seconda delle relazioni.