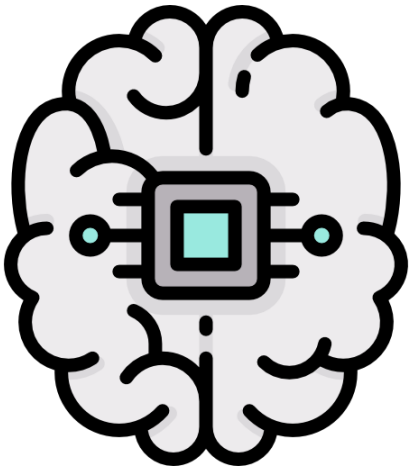

Movie review analyzer

Università degli studi di Napoli "Federico II"
Intelligenza Artificiale
2019-2020

Realizzato da:

Chiara Coppola	*N46004138*
Simone Damiano	*N46004084*
Mariantonietta De Carlo	*N46003688*

INDICE

“Your most unhappy customers are your greatest source of learning.” — Bill Gates

1. Sentiment analysis

1.1) Cos'è la sentiment analysis e quali sono le sue possibili applicazioni

La Sentiment Analysis, conosciuta anche come Opinion Mining, è un campo all'interno del Natural Language Processing (NLP), il cui scopo è l'analisi di un testo con il fine di identificare e classificare l'informazione presente nello stesso. Di solito, oltre a identificare l'opinione, questi sistemi estraggono gli attributi dell'espressione come:

- **Polarità:** opinione positiva o negativa
- **Oggetto:** ciò di cui si parla
- **Opinion holder:** la persona o entità che esprime il parere.

In altri termini l'analisi del sentimento serve per conoscere la brand perception attraverso gli scambi di interazione degli utenti nei social network o più in generale nel web e, grazie all'apprendimento automatico, le aziende sono in grado di estrarre queste opinioni sotto forma di testo o di audio e di analizzare le emozioni che si nascondono dietro di esse su una scala senza precedenti. Tutto questo è possibile perché le informazioni disponibili pubblicamente e privatamente su Internet sono in costante crescita, un gran numero di testi che esprimono opinioni sono disponibili in siti di recensioni, forum, blog e social media.
Con l'analisi dei sentimenti, è possibile trasformare automaticamente queste informazioni non strutturate in dati strutturati di opinioni pubbliche su prodotti, servizi, marchi, politica o qualsiasi argomento su cui le persone possano esprimere opinioni. Questi dati possono essere molto utili per applicazioni commerciali come analisi di marketing, pubbliche relazioni, recensioni di prodotti, punteggio del promotore netto, feedback sul prodotto e servizio clienti.
Si stima che l'80% dei dati nel mondo non sia strutturato e non organizzato in modo predefinito. La maggior parte di questi proviene da dati di testo, come e-mail, ticket di supporto, chat, social media, sondaggi, articoli e documenti. Questi testi sono in genere difficili, lunghi e costosi da analizzare, capire e ordinare.I sistemi di analisi del sentiment consentono alle aziende di dare un senso a questo mare di testo non strutturato automatizzando i processi aziendali, ottenendo informazioni utili e risparmiando ore di elaborazione manuale dei dati, in altre parole, rendendo i team più efficienti.
L'obiettivo principale della web sentiment analysis è 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. È possibile evitare tutto questo e orientare le strategie di comunicazione future esaminando le conversazioni degli utenti sul web (blog, forum, social network).
Alcuni tool e piattaforme consentono di monitorare in real time tutto quello che accade sul web circa un determinato argomento: gli interessati quindi potranno sapere quando un post da positivo, dopo numerose condivisioni diventa negativo, cercando di comprendere le motivazioni arginando la crisi e limitando i danni. Il crisis management viene utilizzato non solo dalle azienda ma anche in politica, specialmente durante le campagne elettorali per tenere monitorato il sentiment legato ai vari partiti o candidati.

Ci sono due approcci generali all'analisi del sentimento:

1) **Pura statistica**: Questi tipi di algoritmi trattano i testi come sacchetti di parole (BOW), dove l'ordine delle parole viene ignorato. Il testo originale viene filtrato fino alle sole parole che si pensa portino un sentimento e sono interessanti nel contesto del discorso. Tali modelli non fanno alcun uso della comprensione di una certa lingua e utilizzano solo misure statistiche per classificare un testo. Questo è l'approccio che abbiamo scelto per il nostro progetto.
2) **Un mix di statistica e linguistica:** Questi algoritmi cercano di incorporare principi grammaticali, varie tecniche di elaborazione del linguaggio naturale e statistiche per addestrare la macchina a 'capire' veramente il linguaggio.

L'analisi del sentimento può anche essere ampiamente classificata in due tipi, in base al tipo di output che l'analisi genera:

1) **Categorico/Polari:** La singola unità del testo era "positivo", "neutro" o "negativo"? In questo processo, si cerca di etichettare un testo come positivo o negativo o neutro.
2) **Scala/Grado:** Viene dato un punteggio su una scala predefinita che va da molto positivo a molto negativo. Ad esempio, la figura seguente mostra un'analisi del sentimento basata su tweet relativi a vari candidati alle elezioni. In questo caso il sentimento viene misurato in forma scalare.

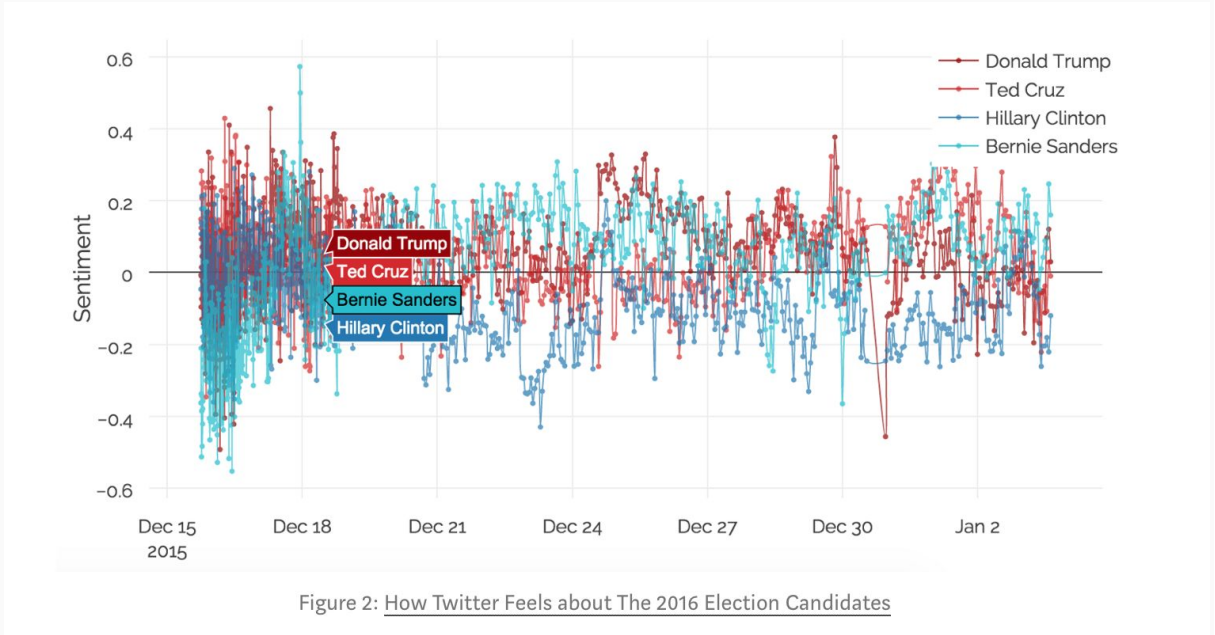

Figure 2: How Twitter Feels about The 2016 Election Candidates

Tuttavia per quanto strabiliante e fruttuosa possa essere questa tecnologia per le aziende (e non solo), ci sono ancora dei limiti e il più grande limite ad oggi ancora non superato è attribuibile al fatto che qualsiasi tool o piattaforma che registra i post e commenti e attribuisce la polarity non è in grado di cogliere concetti emotivi complessi come l'ironia.

Ad esempio un commento che accompagna un'immagine di una reazione cutanea rispetto ad una crema per il corpo che afferma:

"Grazie xxx! Davvero un effetto strepitoso 😕 "

oppure un commento ad un ritardo nel volo:

"Il mio volo è in ritardo. Splendido!"

verranno interpretati e classificati dalla macchina come post dalla polarità positiva mentre invece dovrebbero essere assegnate delle negatività. Quindi in un contesto lessicale pieno di ironia l'attendibilità sarà inferiore rispetto ad un documento con informazioni oggettive. Tale limite viene banalmente superato grazie all'intervento dell'uomo. Altra limitazione è data dall'accuratezza, dalla precisione con cui la macchina ci fornisce i risultati. Parlando di accuratezza del sistema della web sentiment analysis, essa non è altro che il grado di accordo con il giudizio umano.
C'è da precisare che il grado di accuratezza della sentiment analysis migliora all'aumentare del volume di dati (big data), in altre parole milioni di post, possono alleviare le preoccupazioni sull'affidabilità a livello granulare, ossia di un singolo post.

“ When captured electronically, customer sentiment — expressions beyond facts, that convey mood, opinion, and emotion — carries immense business value. We’re talking the voice of the customer, and of the prospect, patient, voter, and opinion leader.” — ***Seth Grimes***

2. Il progetto

2.1) Intro

Il nostro progetto ha l'obiettivo di creare un sistema intelligente in grado di realizzare una sentiment analysis e che, data una recensione testuale, sia in grado di stabilire se quella recensione è positiva o negativa e dirci quanto è sicuro della sua scelta. Questo è possibile grazie al machine learning e alle operazioni di NLP. In particolare con le operazioni di Natural Process Language andiamo a trasformare le nostre recensioni in un qualcosa che sia comprensibile al computer, passando dal dominio del linguaggio a quello dei numeri che sono molto più familiari al calcolatore; queste operazioni comprendono la rimozione della punteggiatura, la riduzione del testo ai soli caratteri minuscoli per non confondere l'IA circa la presenza di nomi propri, la tokenizzazione, l'assegnazione della categoria grammaticale alle parti del discorso ottenute, la scelta di parole che non facciano parte delle stopwords etc. Dei token ottenuti al termine di questa fase selezioniamo solo gli aggettivi partendo dal presupposto che sono gli elementi con più alto contenuto informativo per i nostri scopi. Questa selezioni è necessaria perché le recensioni sono numerose (50.000) e la complessità computazionale sarebbe elevata. Il testo che viene pre-processato proviene da un dataset di recensioni dell'università di Stanford tratte dal sito IMBD ed è suddiviso in recensioni positive e negative. Allenando il nostro sistema applicando diversi modelli, detti classificatori di Bayes (e sue variante) al nostro dataset alla fine saremo in grado di dire, data una recensione, se questa è positiva o negativa e l'accuratezza, la sicurezza con cui la nostra IA afferma il risultato.
L'intero progetto è realizzato in Python e fa uso di alcuni moduli base come os per l'apertura dei file e i moduli NLTK per la pipeline NLP e SKlearn per la parte relativa il machine learning.

2.2) Il codice

Vediamo ora la spiegazione del nostro codice sorgente.
Cominciamo con gli import dei moduli necessari per l'esecuzione del codice. Per ogni libreria saranno date delle informazioni aggiuntive quando verrà spiegata la specifica istruzione.

```
import os
import nltk
import random
import pickle
from statistics import mode
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
import re
```

Il modulo OS del linguaggio Python ha svariate funzioni utili per far compiere diverse operazioni sul sistema operativo del computer come cancellare o rinominare un file, cambiare i permessi in una cartella, trovare l'estensione di un file ecc. In particolare utilizziamo il metodo listdir() che restituisce una lista contenente i nomi delle voci della directory date per percorso, il metodo open() che apre il file il cui nome è specificato in input e il metodo read() che legge dal file. Per il metodo open() 'r' indica la modalità di lettura. Itero l'apertura e la lettura per ogni elemento della lista ottenuta da listdir().

```
files_pos = os.listdir('train/pos')
files_pos = [open('train/pos/' + f, 'r').read() for f in files_pos]
```

All'interno della nostra cartella di progetto abbiamo due cartelle, che rappresentano il training set che diamo in pasto alla nostra IA, con delle recensioni in formato .txt suddivise in recensioni positive e negative. Ripetiamo l'operazione di lettura sia per le recensioni positive che quelle negative.

Tutte le istruzioni da qui in poi vanno a pre-processare il testo per renderlo comprensibile al computer ovvero realizziamo una sequenza di operazioni, detta pipeline NLP, che mi fanno partire da quello che è un testo semi-strutturato o comunque un'informazione non totalmente strutturata e mi fanno arrivare ad un dato che invece è strutturato.
Invochiamo il costruttore della lista per avere una lista delle stopwords, ovvero di quelle parole che possono essere ignorate perché non aggiungono significato alla frase, come gli articoli determinativi e indeterminativi. Con questa istruzione otteniamo così una lista di parole che possono essere rimosse dalle nostre recensioni presenti nel data-set iniziale.

```
stop_words = list(set(stopwords.words('english')))
```

Per i nostri scopi potremmo includere tutte le parole uniche tra quelle da considerare; tuttavia può essere computazionalmente costoso e persino inutile includere tutte le parole uniche nella nostra analisi. Per esempio, il nome di un'attrice in una recensione non darebbe alcuna informazione sul sentimento, sul parere di una recensione quindi dobbiamo essere “smart” e selezionare le parole più informative. Per questo motivo teniamo conto solo degli aggettivi basandoci sul presupposto che gli aggettivi sono altamente informativi sui sentimenti positivi e negativi.

```
all_words = []
documents = []
allowed_word_types = ["J"]
```

In questo ciclo for andiamo di fatto a realizzare le operazioni di “pulizia” e trattamento del testo per tutte le recensioni positive che abbiamo precedentemente estratto dal file e che abbiamo memorizzato nella variabile lista files_pos.
Il metodo append() consente di aggiungere alla nostra lista documents, all’inizio vuota, delle tuple che hanno come primo elemento la recensione, il secondo un’etichetta che in questo caso è “pos” perché stiamo trattando prima le recensioni positive.
Nella variabile cleaned memorizziamo il risultato del metodo sub() che sfrutta il meccanismo delle espressioni regolari per rimuovere la punteggiatura. Il metodo sub() restituisce la stringa ottenuta sostituendo le occorrenze non sovrapposte del primo parametro a sinistra con il secondo parametro all’interno della stringa indicata dal terzo parametro che nel nostro caso è p, la singola recensione.
Le recensioni ripulite dalla punteggiatura vengono tokenizzate in parole con il metodo word_tokenize(). Terminata questa operazione controllo che i token, i singoli pezzettini della mia frase che ho ottenuto non siano contenuti nella lista delle stopwords dopodichè realizzo il POS tagging con l’apposito metodo della libreria NLTK che è pos_tag().
Questo metodo realizza l’analisi grammaticale andando ad assegnare ad ogni token la sua categoria grammaticale. Infine per ogni categoria grammaticale ottenuta, facciamo una lista di aggettivi presenti nella lista di parole consentite definita prima.

```
for p in files_pos:
    documents.append((p, "pos"))
    cleaned = re.sub(r'[^(a-zA-Z)\s]', '', p)
    tokenized = word_tokenize(cleaned)
    stopped = [w for w in tokenized if not w in stop_words]
    pos = nltk.pos_tag(stopped)
    for w in pos:
        if w[1][0] in allowed_word_types:
            all_words.append(w[0].lower())
```

Lo stesso processo viene dualmente ripetuto per le recensioni negative in file_neg.

La lista di aggettivi ottenuta costituisce la nostra Bag Of Words ovvero una borsa di parole (nel nostro caso aggettivi) che dato un testo in input ci consente di ottenerne una rappresentazione matriciale in cui indichiamo la frequenza con cui le parole sono presenti nel testo passando, di fatto, dal dominio del linguaggio a quello dei numeri, decisamente più comprensibile per un computer. Per questo ci appoggiamo alla classe FreqDist ch è usata per codificare le "distribuzioni di frequenza", che contano il numero di volte in cui si verifica ogni risultato e che fa parte delle classi per la rappresentazione e l'elaborazione di informazioni probabilistiche. Operazioni di questo tipo vengono dette analisi lessico-metriche sul testo ovvero analisi basate sul conteggio basico della frequenza delle parole nel testo. Il conteggio delle parole è per noi importante perché quanto più si ripete un aggettivo, maggiore è la sua importanza e quindi il suo significato. La variabile all_words è il nostro BOW finale. Di queste parole realizziamo una lista delle 5000 più frequenti.

```
all_words = nltk.FreqDist(all_words)
word_features = list(all_words.keys())[:5000]
```

Quella che segue è una funzione che crea un dizionario delle caratteristiche per ogni recensione nella lista document. Per ogni recensione creiamo una tupla dove il primo elemento è un dizionario le cui chiavi sono ognuna delle 5000 parole di BOW e i valori per ogni chiave sono o vero se la parola appare nella recensione o falso se la parola non appare. Il secondo elemento è l'etichetta che può essere 'pos' per le recensioni positive e 'neg' per le recensioni negative. Dopodichè la lista di tuple, che da ora in poi prende il nome di feature_set e viene mescolata in maniera casuale con la libreria random, viene suddivisa in training set (da 20,000 elementi) e testing set (da 5,000 elementi).

```
def find_features(document):
    words = word_tokenize(document)
    features = {}
    for w in word_features:
        features[w] = (w in words)
    return features

featuresets = [(find_features(rev), category) for (rev, category) in
documents]

random.shuffle(featuresets)
training_set = featuresets[:20000]
testing_set = featuresets[20000:]
```

Terminata la fase di pre-processo del testo, passiamo ora alla parte di machine learning per apprendere informazioni direttamente dai dati. Da qui in poi comincia un'operazione di classificazione dove utilizziamo il training dataset per ottenere migliori boundary condition che potrebbero essere utilizzate per determinare ogni classe target.
Nel nostro codice abbiamo deciso di combinare la potenza di calcolo di più modelli per ottenere risultati più accurati nella predizione, per questo abbiamo costruito un ensemble model o modello d'insieme che combina i pronostici (ovvero i voti, i risultati ottenuti) di ciascuno di questi modelli per ogni revisione e usa il voto di maggioranza come pronostico finale.
Per costruire il modello d'insieme costruiamo la classe EnsembleClassifier che è inizializzata con una lista di classificatori. È importante che un numero dispari di classificatori sia usato come parte dell'ensemble per evitare la possibilità di un pareggio. La classe ha due metodi principali: classify che restituisce l'etichetta prevista e confidence che restituisce il grado di fiducia nella previsione. Questo grado è misurato come (Numero di voti vincenti)/Votate totali.
Per realizzare ciò importiamo dalla libreria NLTK ClassifierI che è un'interfaccia standard per la "classificazione di una sola categoria", in cui l'insieme delle categorie è noto, il numero di categorie è finito e ogni testo appartiene esattamente ad una categoria (nel nostro caso l'insieme delle categorie è recensione positiva o recensione negativa).

```
from nltk.classify import ClassifierI


class EnsembleClassifier(ClassifierI):
   def __init__(self, *classifiers):
       self._classifiers = classifiers

   def classify(self, features):
       votes = []
       for c in self._classifiers:
           v = c.classify(features)
           votes.append(v)
       return mode(votes)

   def confidence(self, features):
       votes = []
       for c in self._classifiers:
           v = c.classify(features)
           votes.append(v)

       choice_votes = votes.count(mode(votes))
       conf = choice_votes / len(votes)
       return conf
```

Come già affermato prima, per avere dei risultati più accurati classifichiamo utilizzando diversi modelli. Ogni modello può essere allenato con apposite primitive della libreria Sklearn ma l'esecuzione di ognuno è molto lunga per questo memorizziamo tutti i modelli con pickle. Pickle è un modulo di Python che permette di conservare gli oggetti che potrebbero aver impiegato molto tempo ad essere creati prima di chiudere il kernel così da mantenere lo stato del programma in tutte le sessioni.
La funzione da noi definita, load_model(), apre il file indicato dal parametro d'ingresso e carica l'oggetto Pickle, chiude la connessione e ritorna il classificatore che ha appena caricato.
Nell'apprendimento automatico, i classificatori di Bayes sono una famiglia di semplici "classificatori probabilistici" basati sull'applicazione del teorema di Bayes che descrive la probabilità di un evento, sulla base della conoscenza preliminare della condizione. É un metodo popolare di base per la categorizzazione dei testi e il problema di giudicare i documenti come appartenenti a una categoria o all'altra usando le frequenze delle parole come features. A questo combiniamo MNB_Clf che è il classificatore Naive Bayes per modelli multinomiali e BNB_Clf che è il classificatore Naive Bayes simile al MNB_Clf.
Il classificatore multinomiale Naive Bayes è adatto per la classificazione con caratteristiche discrete (ad esempio, conteggio delle parole per la classificazione del testo). Nel classificatore di Bernoulli invece i predittori sono variabili booleane, i parametri che usiamo per predire la variabile di classe occupano solo valori sì o no, per esempio se una parola compare nel testo o meno.
Dopo aver caricato i modelli creiamo un oggetto della nostra classe Ensemble e lo inizializziamo con i nostri modelli e cicliamo per classificare tutte le recensioni contenute nel testing_set.

```
def load_model(file_path):
   classifier_f = open(file_path, "rb")
   classifier = pickle.load(classifier_f)
   classifier_f.close()
   return classifier

ONB_Clf = load_model('pickled_algos/ONB_clf.pickle')

MNB_Clf = load_model('pickled_algos/MNB_clf.pickle')

BNB_Clf = load_model('pickled_algos/BNB_clf.pickle')

ensemble_clf = EnsembleClassifier(ONB_Clf, MNB_Clf, BNB_Clf)

feature_list = [f[0] for f in testing_set]

ensemble_preds = [ensemble_clf.classify(features) for features in
feature_list]
```

Costruita la logica di base, il nostro obiettivo è quello di ottenere una classificazione live di una singola recensione, vogliamo sapere se la nostra macchina è in grado di capire

autonomamente se una recensione è positiva o negativa. Per questo costruiamo una funzione sentiment che dato un testo di una recensione in ingresso costruisce il suo dizionario e ci restituisce in uscita la classificazione che dedotto, ovvero se la recensione è positiva o negativa, e la confidence, ovvero quanto è sicuro della sua predizione.
Data una recensione d'esempio, come text_a, invochiamo la funzione sentiment e stampiamo a video la predizione della nostra IA con il suo grado di confidence in percento.

```
text_a = '''Spider-Man: Homecoming is a disgusting film which re-imagines
the popular Marvel character with little, if any, room for criticism.'''

ris = sentiment(text_a)

if ris[0] == 'pos':
   predizione = "La recensione è positiva al "
else:
   predizione = "La recensione è negativa al "

print(predizione + str(ris[1]*100))
```

---

3. References:

- **Dataset per il training:** *http://ai.stanford.edu/~amaas/data/sentiment/*

- **Cos'è la sentiment analysis:**
  https://www.themarketingfreaks.com/2017/01/cose-la-sentiment-analysis-utilita-limiti-e-tools-gratis-e-a-pagamento/

- **Everything there is to know about sentiment analysis:**
  https://monkeylearn.com/sentiment-analysis/

- **Che cos'è e come funziona la sentiment analysis:**
  http://growthhackingitalia.com/sentiment-analysis/