Tweettalyze

A.Moggio R.Mele R.Pane

IA – CONTEST 2, 02-06-2020

1. Obiettivo

Si vuole creare un applicativo che, dato in ingresso uno o più Tweet, restituisca la loro Sentiment Analysis, nonché una serie di grafici ed informazioni utili riguardanti gli stessi.

2. Possibili Applicazioni

L'obiettivo del sistema è quello di offrire un'analisi sintetica del contenuto dei "Tweet".
Tale analisi offre un insieme di dati che hanno molteplici campi di interesse.
Ad esempio, fornire informazioni circa la percezione degli argomenti di maggiore tendenza, su personaggi di spicco (star, politici, ecc.), o aiutare a definire la Brand Awareness di un determinato marchio.
Insomma, le potenzialità sono praticamente infinite.

3 Let's have a Look Inside

```
[ ]  from colorama import Fore, Style
     from termcolor import colored
     from nltk.tokenize import word_tokenize
     from nltk.stem.wordnet import WordNetLemmatizer
     from nltk.tag import pos_tag
     from nltk.corpus import twitter_samples, stopwords
     from nltk import FreqDist, classify, NaiveBayesClassifier
     from nltk.chat.util import Chat, reflections  # Chatbot
     from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
     from PIL import Image

     import numpy
     import nltk
     #nltk.download('all')
     import re, string, random
     import matplotlib.pyplot as plt
```

In questo primo frammento di codice abbiamo effettuato i diversi **import** per utilizzare in vari strumenti nel programma. Principalmente dal package **nltk** per quanto riguarda l'analisi testuale dei tweet, da **wordcloud** per rappresentare una collezione di parole (estratte dai tweet) in diverse immagini, da **colorama** e **termcolor** per usufruire di diversi colori nella stampa dei risultati e infine **PIL** per l'image processing.

```
[ ]  def pre_process(tweet_tokens, stop_words = ()):

        clean_tok = []

        # Pos-Tag
        for token, tag in pos_tag(tweet_tokens):
            token = re.sub('http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+#]|[!*\(\),]|'\
                           '(?:%[0-9a-fA-F][0-9a-fA-F]))+','', token)
            token = re.sub("(@[A-Za-z0-9_]+)","", token)
            # sub replace the special characters with "" <-- a blank space

            # We generate more generic Tags
            if tag.startswith("NN"):
                pos = 'n'                 # n = Nouns
            elif tag.startswith('VB'):
                pos = 'v'                 # v = Verbs
            else:
                pos = 'a'                 # a = Adjectives and Adverbs

            # Lemmatization
            lemmer = WordNetLemmatizer()           # WordNet is a lexical database for the English Language
            token = lemmer.lemmatize(token, pos)   # Lemmatize

            # we remove the punctuation and lower the characters of the words
            if len(token) > 0 and token not in string.punctuation and token.lower() not in stop_words:
                clean_tok.append(token.lower())
        return clean_tok
```

In questo secondo frammento è definita la funzione **pre_process** che effettua la normalizzazione del testo attraverso le seguenti operazioni:

- Rimozione Stop Words
- Lowering
- Punctuation
- Stemming
- Lemmatization: in particolare è stato utilizzato come database lessicale per la lingua inglese **WordNet**.

Tutte queste fasi realizzano una fase di pre-processing dei vari tweet.

```python
exit_codes=['bye', 'exit', 'see you' ,'ciao']
flag=True
res_lis = []

print("Tweettalyze analyzes your Tweets and makes a Sentimental Analysis of them")
print()

while(flag==True):
    custom_tweet=input("Insert the tweet that you want to analyze:  >>  ")
    print()
    if  custom_tweet.lower() not in exit_codes:
        # We process the tweet. Then we print it(processed) with the result of the classifier(i.e. our Sentment Analysis)
        custom_tokens = pre_process(word_tokenize(custom_tweet)) # (Removing 'Stop Words', Lowering, Punctuation, Stemming , Lemmatization)
        print(custom_tokens,'\n','\n', classifier.classify(dict([token, True] for token in custom_tokens)))
        res_lis.append(classifier.classify(dict([token, True] for token in custom_tokens)))   # it saves the output
        # It asks if the user wants to analyze an other Tweet
        print("\n\n Do you want to analyze a new tweet? (yes o no)")
        print()
        custom_tweet=input()
        print()

        # This is the algorithm that manages the exit point
    if custom_tweet.lower() =="no" or custom_tweet.lower() in exit_codes:
        print("Hi, it was a pleasure chatting with you!")
        flag=False
```

Qui presentiamo il codice del **chat-bot** integrato nel sistema per interagire con l'utente.
Il chat-bot chiede all'utente di inserire i Tweet, uno per volta, attraverso una **text-box**. Una volta inseriti, i Tweet vengono processati secondo gli steps descritti nella fase di pre-processing.
Viene poi effettuata la Sentiment Analysis attraverso il **NaiveBayesClassifier.**
Il risultato sarà quindi o ***Positivo*** o ***Negativo***, a seconda della classificazione fatta sulle componenti del tweet immesso, e quindi dalla sua interpretazione che la macchina gli ha associato.

Fatto ciò, il codice provvede al salvataggio dell'output tramite **append** nella variabile lista **res_lis.** Onde evitare che l'utente resti intrappolato nel ciclo **while,** abbiamo previsto, a fine di ogni inserimento nella text-box e dell'elaborazione, anche la possibilità di uscire tramite delle parole definite nella variabile lista **exit_codes.** L'inserimento di una di queste parole nella text-box pone il flag a **False** e permette di uscire dal ciclo e di terminare l'interazione tra chat-bot e utente.

Di seguito un esempio della console:

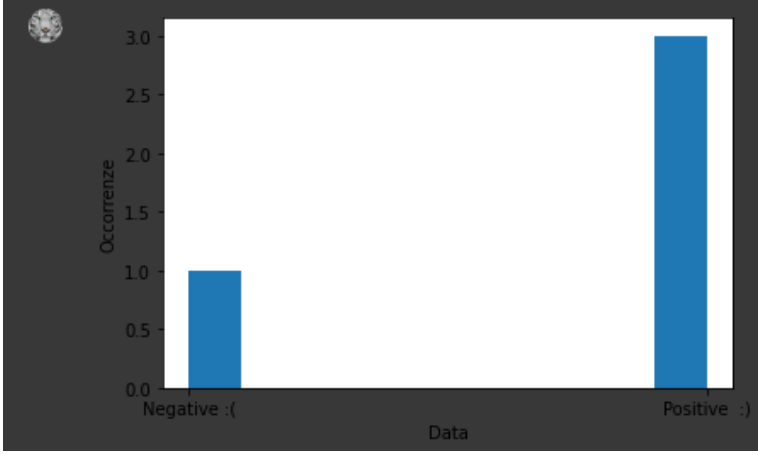
```
Tweettalyze analyzes your Tweets and makes a Sentimental Analysis of them

Insert the tweet that you want to analyze:  >>  I #hate @You!!! ò_ò

['i', 'hate', 'you', 'ò_ò']

 Negative :(
```

```
 Do you want to analyze a new tweet? (yes o no)

no

Hi, it was a pleasure chatting with you!
```

Abbiamo poi importato ed utilizzato la libreria **matplotlib** per la creazione di grafici in Python. In particolare, dopo l'interazione con il chat-bot, all'utente viene resa disponibile la possibilità di visualizzare un grafico; in tale grafico è possibile tenere traccia dei numeri di Tweet **positive** e **negative**.

Di seguito il relativo codice ed una possibile applicazione:

```
[ ]  import matplotlib.pyplot as plt
     import numpy as np
     %matplotlib inline
     x = res_lis

     plt.hist(x, density=False)  # `density=False` would make counts
     plt.ylabel('Occorrenze')
     plt.xlabel('Data');
```

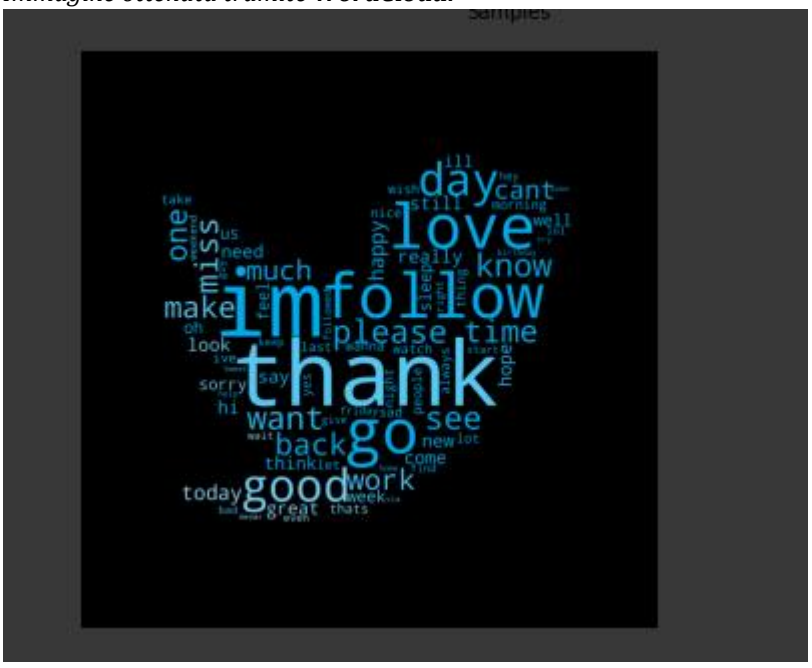


```
# We create the Model for the training and test. For each phase we build a dataset formed by positive and negative shuffled data
positive_data = [(tweet_dict, "Positive  :)")           # This code attaches a Positive label to each tweet
                    for tweet_dict in positive_tokens_for_model]

negative_data = [(tweet_dict, "Negative :(")            # This code attaches a Negative label to each tweet
                    for tweet_dict in negative_tokens_for_model]

dataset = positive_data + negative_data
random.shuffle(dataset)       # it's to avoid Bias

# The purpose of the first part is to build the model; the next part tests the performance of the model.
train_data = dataset[:7500]
test_data = dataset[7500:]

# For the Classification we used a NaiveBayes Classifier
classifier = NaiveBayesClassifier.train(train_data)
# Naive Bayes classifiers are paramaterized by two probability distributions:
#  - P(label) gives the probability that an input will receive each label, given no information about the input's features.
#  - P(fname=fval|label) gives the probability that a given feature (fname) will receive a given value (fval), given that the label (label).

# If the classifier encounters an input with a feature that has never been seen with any label, then rather than assigning a
# probability of 0 to all labels, it will ignore that feature.
```

Il sistema è stato istruito precedentemente nel blocco di codice relativo al Machine Learning. A fine di effettuare la Sentiment Analysis, i Tweet vengono prelevati da un set predefinito della libreria **nltk.** Vengono poi catalogati e salvati nelle variabili **positive_data** e **negative_data**, per poi essere unite in **dataset,** che rappresenta quindi la collezione globale dei nostri dati.
Al fine di allenare il sistema, viene effettuato uno shuffle degli stessi e poi viene effettuata una separazione in **train_data** e **test_data.** Il primo istruisce il sistema su una quantità di dati/Tweet pari al 75% del totale, mente la seconda parte (il restante 25%) serve per testare le performance del modello.

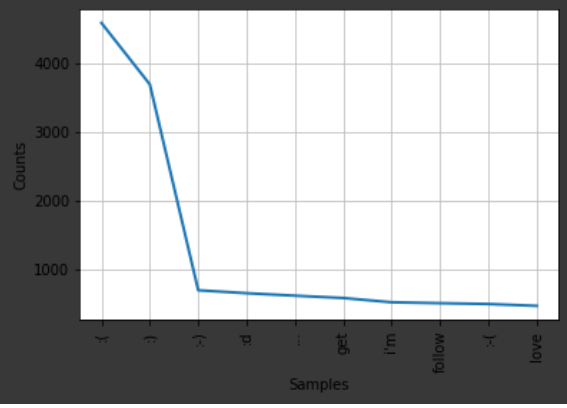
Per effettuare la classificazione si è utilizzato un **NaiveBayesClassifier**, che effettua la clusterizzazione tra positivi e negativi basandosi sull'applicazione del Teorema di Bayes (è un esempio di Reti Bayesiane), e quindi attraverso l'utilizzo di due distribuzioni di probabilità.

*Immagine ottenuta tramite **WordCloud:***



Output della fase di classificazione e resa grafica dei risultati del modello, nonché un coefficiente di Accuratezza:

```
Accuracy is: 0.9964

Most Informative Features
                    :( = True           Negati : Positi =   2236.2 : 1.0
                    :) = True           Positi : Negati =   1062.6 : 1.0
              follower = True           Positi : Negati =     38.1 : 1.0
               welcome = True           Positi : Negati =     21.3 : 1.0
                   bam = True           Positi : Negati =     20.5 : 1.0
                   sad = True           Negati : Positi =     19.2 : 1.0
                    aw = True           Negati : Positi =     16.7 : 1.0
                   ugh = True           Negati : Positi =     14.7 : 1.0
            appreciate = True           Positi : Negati =     14.7 : 1.0
                arrive = True           Positi : Negati =     14.1 : 1.0
None
```



4. So, What's Next?

È possibile espandere il progetto, aggiungendo funzionalità attualmente non sviluppate:

- Migliorare la fase di Machine Learning, tramite un ampliamento del Dataset totale ed un affinamento dei processi di Training Test (ad esempio si potrebbe pensare ad un'analisi complementare basata sui Topic, o l'utilizzo di un classificatore più avanzato)*;*
- Ampliare la quantità di Tweet che si posso analizzare per volta, magari inseriti nel sistema tramite lettura da file;
- Definizioni di ulteriori parametri di raffinamento statistici;
- Implementazione di un'interfaccia più User-Friendly;

5. Strumenti Utilizzati

- Python 3, attraverso Jupyter Notebook e Google Colab
- Overleaf
- Librerie Varie

6. References

- Intelligenza artificiale 3/Ed. - Vol. 1, Un approccio moderno [*Stuart Russell - Peter Norvig*]
- Slides del corso