```
#These are the libraries used for text processing
import pandas as pd #Pandas is  used for data manipulation and analysis
import numpy as np

import nltk #Natural Language Toolkit is used for language processing. This supports stopwords, tokenizing, lemmatizing, and stemming
nltk.download('stopwords') #Nltk stopwords are the most common words and are pre-defined
nltk.download('punkt') #Nltk punkt is used for tokenizaton
nltk.download('wordnet') #Wordnet is used for lemmatization

import spacy #Spacy is used for general purpose natural language processing

import re #Re is used for supporting regular expression

from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer #Used for word lemmatization
from nltk.stem import PorterStemmer #Used for word stemming

import unicodedata #Unicodedata library grants properties for Unicode characters from the Unicode Character Database

import string #String library support string manipulation

from textblob import TextBlob #TextBlob library is used for nlp tasks

!pip install transformers
import transformers #Transformers library provides a large selection of models and tools for nlp.
from transformers import pipeline #High-level API that simplifies the process of using pre-trained models for various NLP tasks.

import warnings
warnings.simplefilter(action='ignore', category=FutureWarning) #Ignores future warnings in code
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.30.1)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers) (3.12.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.14.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.15.1)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (1.22.4)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers) (2022.10.31)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers) (2.27.1)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.13.3)
Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers) (0.3.1)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers) (4.65.0)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1->transformers) (2023
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1-
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (2.0.1
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers) (3.4)
```

```
#Import the csv file of the dataset. The dataset comprises of tweets from Bill Gates
url = 'https://raw.githubusercontent.com/jantupas/dsfa2/main/billgates.csv'

df = pd.read_csv(url,encoding='windows-1252')

df.loc[:,'Text']
```

```
0     In 1995, I wrote about “the internet tidal wav...
1     The world has made incredible progress against...
2     Vaccines work to #EndPolio and protect communi...
3     .@bjornlomborg and I agree—the Global Goals ar...
4     Here’s what I’m listening to this summer. Hope...
5     I hope you have plenty of content to enjoy thi...
6     I’m excited by the positive momentum around th...
7     For the past decade, I’ve recommended books to...
8     RT @gatesfoundation: The transition to high sc...
9     It was an honor to meet Navajo Nation Presiden...
10    RT @MSuzman: For @GatesFoundation, our focus i...
11    Really good news: Renewables now account for a...
12    RT @gatesfoundation: Still today, your zip cod...
13    I often say Richard Feynman is the best teache...
14    My message to the class of 2023. https://t.co/...
```

```
15    Ric was employee number 2 at Microsoft and a d...
16    RT @AlzData: Applications are now open for the...
17    Building a clean energy future will require a ...
18    It’s possible to build nuclear reactors withou...
19    I had the best day in Kemmerer, Wyoming. Over ...
20    Expanding access to single-dose HPV vaccines c...
21    I’m very encouraged by the E-MOTIVE Trial resu...
22    By prioritizing accessibility, affordability, ...
23    Eradicating smallpox was hard, but it was also...
24    Nuclear energy, if we do it right, will help u...
25    Aside from water, there’s no material on Earth...
26    This next-generation nuclear facility will be ...
27    On International Day of the Midwife, I’m think...
28    RT @gatesfoundation: Is college the right choi...
29    When I first met @dsengeh in 2011, I was blown...
30    As we rely more and more on electricity, it's ...
31    Mann ki Baat has catalyzed community led actio...
32    To #EndPolio, we must immunize the world's mos...
33    The connection is clear: when vaccine coverage...
34    This new fellowship powered by the AD Data Ini...
35    Everyone should have the same goal in mind: en...
36    RT @MalariaNoMore: Today is #WorldMalariaDay!\...
37    Despite being both preventable and curable, ma...
38    This new discovery in mathematics is really co...
39    Uganda’s maternal mortality rate is at least d...
40    They say the eyes are the windows to the soul,...
41    You might see me using the “bathtub metaphor” ...
42    RT @gatesfoundation: Chickpeas? Garbanzo beans...
43    This week in In San Diego, I got to talk with ...
44    For decades, we’ve been fighting back against ...
45    It’s always great catching up with @jessieww. ...
46    I had a lot of fun meeting @CheechMarin, exper...
47    I am so impressed with Eva Nangalo—it’s hard n...
48    I’m grateful for the Lauder family’s dedicatio...
49    I’m excited to see @devisridhar‘s book in pape...
Name: Text, dtype: object
```

Show Original Text

```
#Shows the tweets of Bill Gates in original text. Values in the text column from the data frame df are transferred to the list dataset.
dataset = df.loc[:,'Text'].tolist()
print(dataset)
```

```
['In 1995, I wrote about “the internet tidal wave.” I even mentioned how the CD-ROM business would be dramatically impacted by the Inter
```

Lower Casing Text

```
#In analyzing the dataset, the presence of uppercase characters showed no significance. Lower casing allows us to easily process text and han
df['Lower_Text'] = df['Text'].str.lower() #all strings under the text column are converted to lower case. They are stored in a new column nam

#Shows the text that are now in Lower Case. Values from df Lower_Text column replace the previous values in the list dataset.
dataset = df.loc[:,'Lower_Text'].tolist()
print(dataset)
```

```
['in 1995, i wrote about “the internet tidal wave.” i even mentioned how the cd-rom business would be dramatically impacted by the inter
```

Punctuation, Target Mentions, and Hashtag Removal

```
#Removing punctuation marks, target mentions and hashtags lessens noise in the dataset. We use regular expression and str.replace method to m
df['Clean_Text_A'] = df['Lower_Text'].str.replace ('@[A-Za-z0-9_]+','') #Removes all words that begin with @. @ signifies a target mention. T
df['Clean_Text_A'] = df['Clean_Text_A'].str.replace ('#[A-Za-z0-9_]+','') #Removes all words that begin with #. # signifies a hashtage
df['Clean_Text_A'] = df['Clean_Text_A'].str.replace ('[^\w\s]','') #Removes all characters that do not match Unicode word characters.

#Shows the text without punctuation marks, target mentions, and hashtags. Values from df Clean_Text_A column replace the previous values in t
dataset = df.loc[:,'Clean_Text_A'].tolist()
print(dataset)
```

```
['in 1995 i wrote about the internet tidal wave i even mentioned how the cdrom business would be dramatically impacted by the internet i
```

URL, Newline, and Re-tweets Removal

```
#Removing url, newlines and re-tweets further lessens noise in the dataset. Re-tweets are not original text posted by the user which causes s
```

```
df['Clean_Text_B'] = df['Clean_Text_A'].str.replace('https[A-Za-z0-9_]+','') #Removes all words that begin with https. https signifies a url.
df['Clean_Text_B'] = df['Clean_Text_B'].str.replace('\n','') #Removes all \n characters. \n signifies newline.
df.drop(df[df['Clean_Text_B'].str.startswith('rt')].index, inplace = True) #Identifies all texts that begin with rt and removes the entire ro
df.reset_index(inplace = True) #Reset the index because of the rows Clean_Text_B.

#Shows the text without urls, newlines, and are not re-tweets. Values from df Clean_Text_B column replace the previous values in the list dat
dataset = df.loc[:,'Clean_Text_B'].tolist()
print(dataset)
```

```
['in 1995 i wrote about the internet tidal wave i even mentioned how the cdrom business would be dramatically impacted by the internet i
```

Stop Words Removal

```
#Stop words are common, low information words and should be removed to further reduce noise and sparity. We use all nltk pre-defined stop wor
stop_words = stopwords.words('english') #all pre-defined stopwords are stored in list stop_words

#remove_stop function runs through all words in text and joins only words that are not in list stop_words.
def remove_stop(x):
  return ' '.join([word for word in str(x).split() if word not in stop_words])

#remove_stop function is applied in the values of column Clean_Text_B. Results are stored in a new column called Clean_Text_C.
df['Clean_Text_C'] = df['Clean_Text_B'].apply(lambda x: remove_stop(x))

#Shows the text without stop words. Values from df Clean_Text_C column replace the previous values in the list dataset.
dataset = df.loc[:,'Clean_Text_C'].tolist()
print(dataset)
```

```
['1995 wrote internet tidal wave even mentioned cdrom business would dramatically impacted internet fun look back think far technology c
```

Numerical Data Removal

```
#In the chosen dataset, numerical data do not signify keywords necessary for nlp resulting in further noise and sparity. We use regular expre
df['No_Num_Text'] = df['Clean_Text_C'].str.replace('[0-9]','') #Removes all numbers. The texts are stored in a new column named No_Num_Text

#Shows the text without numerical data. Values from df No_Num_Text column replace the previous values in the list dataset.
dataset = df.loc[:,'No_Num_Text'].tolist()
print(dataset)
```

```
[' wrote internet tidal wave even mentioned cdrom business would dramatically impacted internet fun look back think far technology come
```

White Space Removal

```
#White spaces should be removed from text to improve and simplify text preprocessing. We use regular expression to match and replace all whit
df['No_White_Text'] = df['No_Num_Text'].str.replace(' +',' ').str.strip() #Removes all white spaces. The texts are stored in a new column nam

#Shows the text without white spaces. Values from df No_White_Text column replace the previous values in the list dataset.
dataset = df.loc[:,'No_White_Text'].tolist()
print(dataset)
```

```
['wrote internet tidal wave even mentioned cdrom business would dramatically impacted internet fun look back think far technology come',
```

Tokenization

```
#Tokenization splits text into tokens which are individual words to easily assign meaning when used for nlp. We use re.split method and regul

#We apply re.split method to df column No_White_Text to tokenize its values. The results are stored in a new column named Word_Token
df['Word_Token'] = df['No_White_Text'].apply(lambda x: re.split('\W+',x))

#Shows the text that has been tokenized. Values from df Word_Token column replace the previous values in the list dataset.
dataset = df.loc[:,'Word_Token'].tolist()
print(dataset)
```

```
[['wrote', 'internet', 'tidal', 'wave', 'even', 'mentioned', 'cdrom', 'business', 'would', 'dramatically', 'impacted', 'internet', 'fun
```

Lemmatization

```
#Lemmatization reduces a word to its root form and meaning to improve the process of nlp. We use WordNetLemmatizer method from nltk to lemmat

#lemmatize_word function uses WordNetLemmatizer method to lemmatize the word and stores it into list lemmatized_word. When the function is ca
def lemmatize_word(txt):
  lemmatized_word = [WordNetLemmatizer().lemmatize(word) for word in txt]
  return lemmatized_word

#We apply lemmatize_word function to the values of df column Word_Token. The results are stored in a new column named Word_Lemmatize
df['Word_Lemmatize'] = df['Word_Token'].apply(lambda x: lemmatize_word(x))

#Shows the text that has been lemmatized. Values from df Word_Lemmatize column replace the previous values in the list dataset.
dataset = df.loc[:,'Word_Lemmatize'].tolist()
print(dataset)
```

```
[['wrote', 'internet', 'tidal', 'wave', 'even', 'mentioned', 'cdrom', 'business', 'would', 'dramatically', 'impacted', 'internet', 'fun
```

Stemming

```
#Stemming reduces the inflection of a word to their root form to support and improve text pre-processing and nlp. We use PorterStemmer method

#stem_word function used PorterStemmer method to stem the word and stores it into list stemmed_word. When the function is called, the list st
def stem_word(txt):
  stemmed_word = [PorterStemmer().stem(word) for word in txt]
  return stemmed_word

#We apply stem_wordfunction to the values of df column Word_Lemmatize. The results are stored in a new column named Word_Stem
df['Word_Stem'] = df['Word_Lemmatize'].apply(lambda x: stem_word(x))

#Shows the text that has been stemmed. Values from df Word_Stem column replace the previous values in the list dataset.
dataset = df.loc[:,'Word_Stem'].tolist()
print(dataset[0:20])
```

```
[['wrote', 'internet', 'tidal', 'wave', 'even', 'mention', 'cdrom', 'busi', 'would', 'dramat', 'impact', 'internet', 'fun', 'look', 'bac
```

```
#Processed Tweets
df.loc[:,'Word_Stem']
```

```
0     [wrote, internet, tidal, wave, even, mention, ...
1     [world, made, incred, progress, polio, infecti...
2     [vaccin, work, protect, commun, dozen, diseas,...
3     [agreeth, global, goal, phenomen, idea, perfec...
4     [here, im, listen, summer, hope, youll, think,...
5     [hope, plenti, content, enjoy, summer, your, l...
6     [im, excit, posit, momentum, around, global, h...
7     [past, decad, ive, recommend, book, read, summ...
8     [honor, meet, navajo, nation, presid, first, l...
9     [realli, good, news, renew, account, one, thir...
10    [often, say, richard, feynman, best, teacher, ...
11                                    [messag, class]
12    [ric, employe, number, microsoft, dear, friend...
13    [build, clean, energi, futur, requir, lot, new...
14    [possibl, build, nuclear, reactor, without, en...
15    [best, day, kemmer, wyom, coal, plant, tour, s...
16    [expand, access, singledos, hpv, vaccin, could...
17    [im, encourag, emot, trial, result, releas, to...
18    [priorit, access, afford, econom, mobil, redef...
19    [erad, smallpox, hard, also, one, import, achi...
20    [nuclear, energi, right, help, u, solv, climat...
21    [asid, water, there, materi, earth, use, concr...
22    [nextgener, nuclear, facil, win, local, econom...
23    [intern, day, midwif, im, think, eva, nangaloa...
24    [first, met, blown, away, intellect, ambit, se...
25    [reli, electr, vital, find, innov, way, gener,...
26    [mann, ki, baat, catalyz, commun, led, action,...
27    [must, immun, world, vulner, childrenespeci, v...
28    [connect, clear, vaccin, coverag, go, see, out...
29    [new, fellowship, power, ad, data, initi, aim,...
30    [everyon, goal, mind, end, prevent, diseas, li...
31    [despit, prevent, curabl, malaria, still, clai...
32             [new, discoveri, mathemat, realli, cool]
33    [uganda, matern, mortal, rate, least, doubl, g...
34    [say, eye, window, soul, could, also, key, unl...
35    [might, see, use, bathtub, metaphor, next, tim...
36    [week, san, diego, got, talk, issu, today, mat...
37    [decad, weve, fight, back, deadli, diseas, exp...
38    [alway, great, catch, chat, math, edtech, ai, ...
```

```
39    [lot, fun, meet, experienc, learn, lot, work, ...
40    [impress, eva, nangaloit, hard, she, spent, de...
41    [im, grate, lauder, famili, dedic, solv, alzheim]
42    [im, excit, see, book, paperback, way, think, ...
Name: Word_Stem, dtype: object
```

Applying Sentiment Analysis Using A RoBERTa-base Model

```
#Here we used a RoBERTa-base Model for sentiment analysis that has been trained on 124 million tweets. The model was retrieved from Hugging F

#We download the RoBERTa-base sentiment analysis model from hugging face. The model is applied to the dataset
dataset = df.loc[:,'No_White_Text'].tolist()
print(dataset)
sentiment_model = pipeline(model ='cardiffnlp/twitter-roberta-base-sentiment')

#Show the results of the sentiment analysis. LABEL_0 means NEGATIVE, LABEL_1 means NEUTRAL, LABEL_2 means POSITIVE.
sentiment_model(dataset)
```

```
['wrote internet tidal wave even mentioned cdrom business would dramatically impacted internet fun look back think far technology come',
Xformers is not installed correctly. If you want to use memory_efficient_attention to accelerate training use the following command to i
pip install xformers.
[{'label': 'LABEL_2', 'score': 0.6453714370727539},
 {'label': 'LABEL_2', 'score': 0.8911813497543335},
 {'label': 'LABEL_1', 'score': 0.49357324838638306},
 {'label': 'LABEL_2', 'score': 0.7566262483596802},
 {'label': 'LABEL_1', 'score': 0.5774592757225037},
 {'label': 'LABEL_2', 'score': 0.956655740737915},
 {'label': 'LABEL_2', 'score': 0.8767237067222595},
 {'label': 'LABEL_1', 'score': 0.7798121571540833},
 {'label': 'LABEL_1', 'score': 0.6674532294273376},
 {'label': 'LABEL_2', 'score': 0.955295979976654},
 {'label': 'LABEL_2', 'score': 0.7240356802940369},
 {'label': 'LABEL_1', 'score': 0.6101581454277039},
 {'label': 'LABEL_2', 'score': 0.7944564819335938},
 {'label': 'LABEL_1', 'score': 0.5714358687400818},
 {'label': 'LABEL_1', 'score': 0.5473719835281372},
 {'label': 'LABEL_2', 'score': 0.9616201519966125},
 {'label': 'LABEL_2', 'score': 0.9469780325889587},
 {'label': 'LABEL_2', 'score': 0.8445103764533997},
 {'label': 'LABEL_2', 'score': 0.9702820181846619},
 {'label': 'LABEL_2', 'score': 0.785840630531311},
 {'label': 'LABEL_2', 'score': 0.9252800345420837},
 {'label': 'LABEL_1', 'score': 0.792206883430481},
 {'label': 'LABEL_1', 'score': 0.7244055867195129},
 {'label': 'LABEL_1', 'score': 0.8023517727851868},
 {'label': 'LABEL_2', 'score': 0.9030808806419373},
 {'label': 'LABEL_2', 'score': 0.4986259639263153},
 {'label': 'LABEL_2', 'score': 0.7839193344116211},
 {'label': 'LABEL_2', 'score': 0.48219624161720276},
 {'label': 'LABEL_1', 'score': 0.5723499059677124},
 {'label': 'LABEL_2', 'score': 0.6040951013565063},
 {'label': 'LABEL_2', 'score': 0.6617953181266785},
 {'label': 'LABEL_2', 'score': 0.6807810068130493},
 {'label': 'LABEL_2', 'score': 0.9781891703605652},
 {'label': 'LABEL_1', 'score': 0.6223629713058472},
 {'label': 'LABEL_1', 'score': 0.8490437269210815},
 {'label': 'LABEL_2', 'score': 0.6170380711555481},
 {'label': 'LABEL_1', 'score': 0.8687952160835266},
 {'label': 'LABEL_2', 'score': 0.5615702867507935},
 {'label': 'LABEL_2', 'score': 0.9616062641143799},
 {'label': 'LABEL_2', 'score': 0.9365656971931458},
 {'label': 'LABEL_2', 'score': 0.5762225985527039},
 {'label': 'LABEL_2', 'score': 0.8900753855705261},
 {'label': 'LABEL_2', 'score': 0.9728915095329285}]
```

Applying Sentiment Analysis Using A TextBlob

```
#Here we used TextBlob for sentiment analysis. It is a library for nlp. We use subjectivity and polarity for analysis.

#Function to get the subjectivity of a tweet
def getSubjectivity(text):
  return TextBlob(text).sentiment.subjectivity

#Function to get the polarity of a tweet.
def getPolarity(text):
  return TextBlob(text).sentiment.subjectivity

#Functions are applied to the dataset.
```

```
#Functions are applied to the dataset.
df['Subjectivity'] = df['No_White_Text'].apply(getSubjectivity)
df['Polarity'] = df['No_White_Text'].apply(getPolarity)

#Function to analyze tweets based on polarity. A score < 0 is negative, score == 0 is neutral, score > 0 is positive
def getAnalysis(score):
  if score < 0:
    return 'Negative'
  elif score == 0:
    return 'Neutral'
  else:
    return 'Positive'

#Analysis function is applied to the dataset
df['Analysis'] = df['Polarity'].apply(getAnalysis)

#Show the results sentiment analysis.
df.loc[:,['Subjectivity','Polarity','Analysis']]
```

	Subjectivity	Polarity	Analysis
0	0.400000	0.400000	Positive
1	0.900000	0.900000	Positive
2	0.000000	0.000000	Neutral
3	0.525000	0.525000	Positive
4	0.666667	0.666667	Positive
5	0.500000	0.500000	Positive
6	0.413636	0.413636	Positive
7	0.250000	0.250000	Positive
8	0.333333	0.333333	Positive
9	0.200000	0.200000	Positive
10	0.650000	0.650000	Positive
11	0.000000	0.000000	Neutral
12	0.300000	0.300000	Positive
13	0.335909	0.335909	Positive
14	0.866667	0.866667	Positive
15	0.468750	0.468750	Positive
16	0.218750	0.218750	Positive
17	0.227273	0.227273	Positive
18	0.550000	0.550000	Positive
19	0.435417	0.435417	Positive
20	0.495238	0.495238	Positive
21	0.650000	0.650000	Positive

✓ 0s completed at 12:51 AM

Analysis

The dataset comprised tweets from Bill Gates. The tweets were gathered using RapidMiner. Pre-processing techniques were applied to the dataset to lessen as much noise and sparsity as possible. The techniques used were the following: Lower Casing Text; Punctuation, Target Mentions, and Hashtag Removal; URL, Newline, Re-Tweets Removal; Stop Words Removal; Numerical Data Removal; White Space Removal; Tokenization; Lemmatization; Stemming. Two models were used to carry out sentiment analysis on the data set.

The first model used was a RoBERTa-base Model for sentiment analysis. The model was retrieved from Hugging Face and has been trained on 124 million tweets. The results from the model show that the sentiment of the tweets falls on the positive and neutral level, where 67.44% of the tweets were positive, and 32.67% of the tweets were neutral. No tweets were associated with a negative sentiment. RoBERTa-base is a variant of the RoBERTa (Robustly Optimized BERT) model, a transformer-based neural network model. The model is refined on a labeled dataset containing text samples annotated with sentiment labels to perform sentiment analysis. RoBERTa-base can discern the mood conveyed in a text and generate predictions in line with that understanding. Using the contextual embeddings pre-trained during the pre-training, which involved 124 million tweets, text can be categorized into sentiment categories or even given a sentiment score that indicates the strength of the communicated sentiment. This signifies why the model produced the sentiment scores of the tweets from the dataset.

The second model used was TextBlob which evaluates the text and produces a polarity score. The polarity is evaluated as either positive, negative, or neutral sentiment. The model identifies positive and neutral as the sentiments of the tweets, where 90.70% of the tweets were positive, and 9.30% of the tweets were neutral. Using a sentiment lexicon and a machine learning model that has already been trained, TextBlob determines sentiment ratings for sentiment analysis. Each word in the lexicon has a polarity score representing the attitude or feeling it is linked with. TextBlob's sentiment analysis method is relatively straightforward in comparison to more complex algorithms. While it offers a speedy sentiment analysis solution, it may not accurately capture subtle subtleties or context-specific sentiment.

The differences in the percentage of positive and neutral sentiment in the dataset between the two models are significant. This is attributed to the training that each model has gone through, where the RoBERT-a model was specifically trained with datasets of tweets. This implies that the RoBERTa-base model provides more accurate sentiment scores than TextBlob for the sentiment analysis of tweets.