**Bahria University,**

**Karachi Campus**



## LAB EXPERIMENT NO.

6

## LIST OF TASKS

|  |  |
| --- | --- |
| **TASK NO** | **OBJECTIVE** |
| **1** | Using python implements VADER rulesbased classification algorithm to find the sentiments of different sentences. |
| **2** | :Using python implements textBlob rules-based classification algorithm to find the sentiments of different sentences and compare the results with task # 01. |

Submitted On:

25 feb 2024

(Date: DD/MM/YY)

**TASK # 1:** Using python implements VADER rulesbased classification algorithm to find the sentiments of different sentences.

import nltk

from nltk.sentiment import SentimentIntensityAnalyzer

import matplotlib.pyplot as plt

sid = SentimentIntensityAnalyzer()

def analyze\_sentiment\_and\_generate\_pie\_chart(sentence):

    scores = sid.polarity\_scores(sentence)

    compound\_score = scores['compound']

    if compound\_score >= 0.05:

        sentiment = "Positive"

    elif compound\_score <= -0.05:

        sentiment = "Negative"

    else:

        sentiment = "Neutral"

    print(f"Sentence: {sentence}\nScores: {scores}\nSentiment: {sentiment}\n")

    labels = ['Positive', 'Neutral', 'Negative']

    sizes = [scores['pos'], scores['neu'], scores['neg']]

    colors = ['yellowgreen', 'gold', 'lightcoral']

    explode = (0.1, 0, 0)

    plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)

    plt.axis('equal')

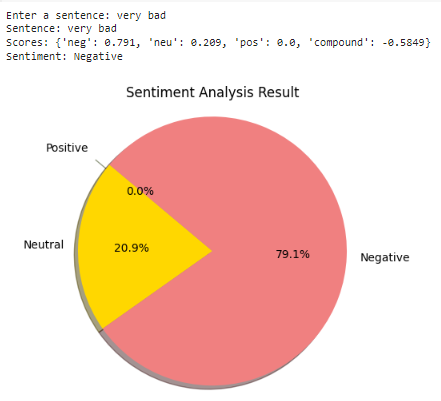
    plt.title('Sentiment Analysis Result')

    plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    sentence = input("Enter a sentence: ")

    analyze\_sentiment\_and\_generate\_pie\_chart(sentence)



**TASK # 2:** Using python implements textBlob rules-based classification algorithm to find the sentiments of different sentences and compare the results with task # 01

import matplotlib.pyplot as plt

from textblob import TextBlob

import nltk

from nltk.sentiment import SentimentIntensityAnalyzer

def get\_textblob\_sentiment(sentence):

    analysis = TextBlob(sentence)

    return analysis.sentiment.polarity

def get\_vader\_sentiment(sentence):

    sid = SentimentIntensityAnalyzer()

    scores = sid.polarity\_scores(sentence)

    return scores['compound']

def display\_comparison\_pie\_chart(textblob\_polarity, vader\_compound):

    labels = ['TextBlob', 'VADER']

    sizes = [textblob\_polarity, vader\_compound]

    colors = ['lightblue', 'lightgreen']

    explode\_distance = 0.1

    explode = (explode\_distance if textblob\_polarity < 0 else 0, explode\_distance if vader\_compound < 0 else 0)

    plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%',

            shadow=True, startangle=90)

    plt.axis('equal')

    plt.title('Sentiment Analysis Comparison')

    plt.show()

if \_\_name\_\_ == "\_\_main\_\_":

    sentence = input("Enter a sentence for sentiment analysis: ")

    textblob\_polarity = get\_textblob\_sentiment(sentence)

    vader\_compound = get\_vader\_sentiment(sentence)

    print(f"TextBlob Sentiment Polarity: {textblob\_polarity}\nVADER Sentiment Compound: {vader\_compound}")

    textblob\_pie\_value = (textblob\_polarity + 1) \* 50

    vader\_pie\_value = (vader\_compound + 1) \* 50

    display\_comparison\_pie\_chart(textblob\_pie\_value, vader\_pie\_value)

