```
 1 import time
 2 import pandas as pd
 3 import numpy as np
 4 import matplotlib.pyplot as plt
 5 from tkinter import *
 6 import tkinter.messagebox
 7 from nltk.sentiment.vader import
   SentimentIntensityAnalyzer
 8 from vaderSentiment.vaderSentiment import
   SentimentIntensityAnalyzer
 9
10
11 class analysis_text():
12
13     # Main function in program
14     def center(self, toplevel):
15
16         toplevel.update_idletasks()
17         w = toplevel.winfo_screenwidth()
18         h = toplevel.winfo_screenheight()
19         size = tuple(int(_) for _ in
20                      toplevel.geometry().
   split('+')[0].split('x'))
21
22         x = w / 2 - size[0] / 2
23         y = h / 2 - size[1] / 2
24         toplevel.geometry("%dx%d+%d+%d"
    % (size + (x, y)))
25
26     def callback(self):
27         if tkinter.messagebox.askokcancel(
   "Quit",
28
```

```
28 "Do you want to leave?"):
29             self.main.destroy()
30 
31     def setResult(self, type, res):
32 
33         # calculated comments in vader
   analysis
34         if (type == "neg"):
35             self.negativeLabel.configure(
   text=
36                                          "
   you typed negative comment : "
37                                          +
    str(res) + " % \n")
38         elif (type == "neu"):
39             self.neutralLabel.configure(
   text=
40                                         "
   you typed  comment : "
41                                         +
   str(res) + " % \n")
42         elif (type == "pos"):
43             self.positiveLabel.configure(
   text
44                                          =
   "you typed positive comment: "
45 
   + str(res) + " % \n")
46 
47 
48     def runAnalysis(self):
49 
50         sentences = []
```

```
51          sentences.append(self.line.get())
52          sid = SentimentIntensityAnalyzer()
53
54          for sentence in sentences:
55
56              # print(sentence)
57              ss = sid.polarity_scores(
   sentence)
58
59              if ss['compound'] >= 0.05:
60                  self.normalLabel.configure
   (text=
61
    " you typed positive statement: ")
62
63              elif ss['compound'] <= - 0.05:
64                  self.normalLabel.configure
   (text=
65
    " you typed negative statement")
66
67              else:
68                  self.normalLabel.configure
   (text=
69
    " you normal typed  statement: ")
70              for k in sorted(ss):
71                  self.setResult(k, ss[k])
72          print()
73
74      def editedText(self, event):
75          self.typedText.configure(text=self
   .line.get() + event.char)
```

```
 76
 77      def runByEnter(self, event):
 78          self.runAnalysis()
 79
 80      def __init__(self):
 81          # Create main window
 82          self.main = Tk()
 83          self.main.title("Text Detector &
    Analysis system  ")
 84          self.main.geometry("700x800")
 85          self.main.resizable(width=FALSE,
    height=FALSE)
 86          self.main.protocol("
    WM_DELETE_WINDOW", self.callback)
 87          self.main.focus()
 88          self.center(self.main)
 89
 90          # addition item on window
 91          self.label1 = Label(text="type a
    text here :")
 92          self.label1.pack()
 93
 94          # Add a hidden button Enter
 95          self.line = Entry(self.main,
    width=70)
 96          self.line.pack()
 97
 98          self.textLabel = Label(text="\n",
 99                                 font=("
    Helvetica", 15))
100          self.textLabel.pack()
101          self.typedText = Label(text="",
102                                 fg="blue",
```

```
103                                     font=("
    Helvetica", 20))
104         self.typedText.pack()
105
106         self.line.bind("<Key>", self.
    editedText)
107         self.line.bind("<Return>", self.
    runByEnter)
108
109         self.result = Label(text="\n",
110                             font=("
    Helvetica", 15))
111         self.result.pack()
112         self.negativeLabel = Label(text=
    "",
113                                    fg="
    red",
114                                    font=(
    "Helvetica", 20))
115         self.negativeLabel.pack()
116         self.neutralLabel = Label(text=""
    ,
117                                   font=("
    Helvetica", 20))
118         self.neutralLabel.pack()
119         self.positiveLabel = Label(text=
    "",
120                                    fg="
    green",
121                                    font=(
    "Helvetica", 20))
122         self.positiveLabel.pack()
123         self.normalLabel = Label(text="",
```

```
124                                                 fg="red"
    ,
125                                                 font=("
    Helvetica", 20))
126             self.normalLabel.pack()
127
128
129 # Driver code
130 myanalysis = analysis_text()
131 mainloop()
132
```