SUMMER TRAINING REPORT

On

“Sentimental analysis”

Submitted by

Ashree Verma

171520007

Department of Computer Engineering & Applications

Institute of Engineering & Technology

GLA University

Mathura- 281406, INDIA

2019

Department of computer Engineering and Applications
GLA University, Mathura
17 km. Stone NH#2, Mathura-Delhi Road, P.O. – Chaumuhan, Mathura – 281406

Declaration

I hereby declare that the work which is being presented in the Summer Training project "**sentimental analysis**"**,** in partial fulfillment of the requirements for Summer Training viva voce, is an authentic record of my own work carried under the supervision of "UPTEC Computer Consultansy Ltd Kanpur".

Signature of Candidate:

Name of Candidate: Ashree Verma

Roll. No. : 171520007

Course: B.Tech (DA)

Year: 3rd

Semester: V

CONTENTS

Department of computer Engineering and Applications GLA University, Mathura

17 km. Stone NH-2, Mathura-Delhi Road, P.O. – Chaumuhan,

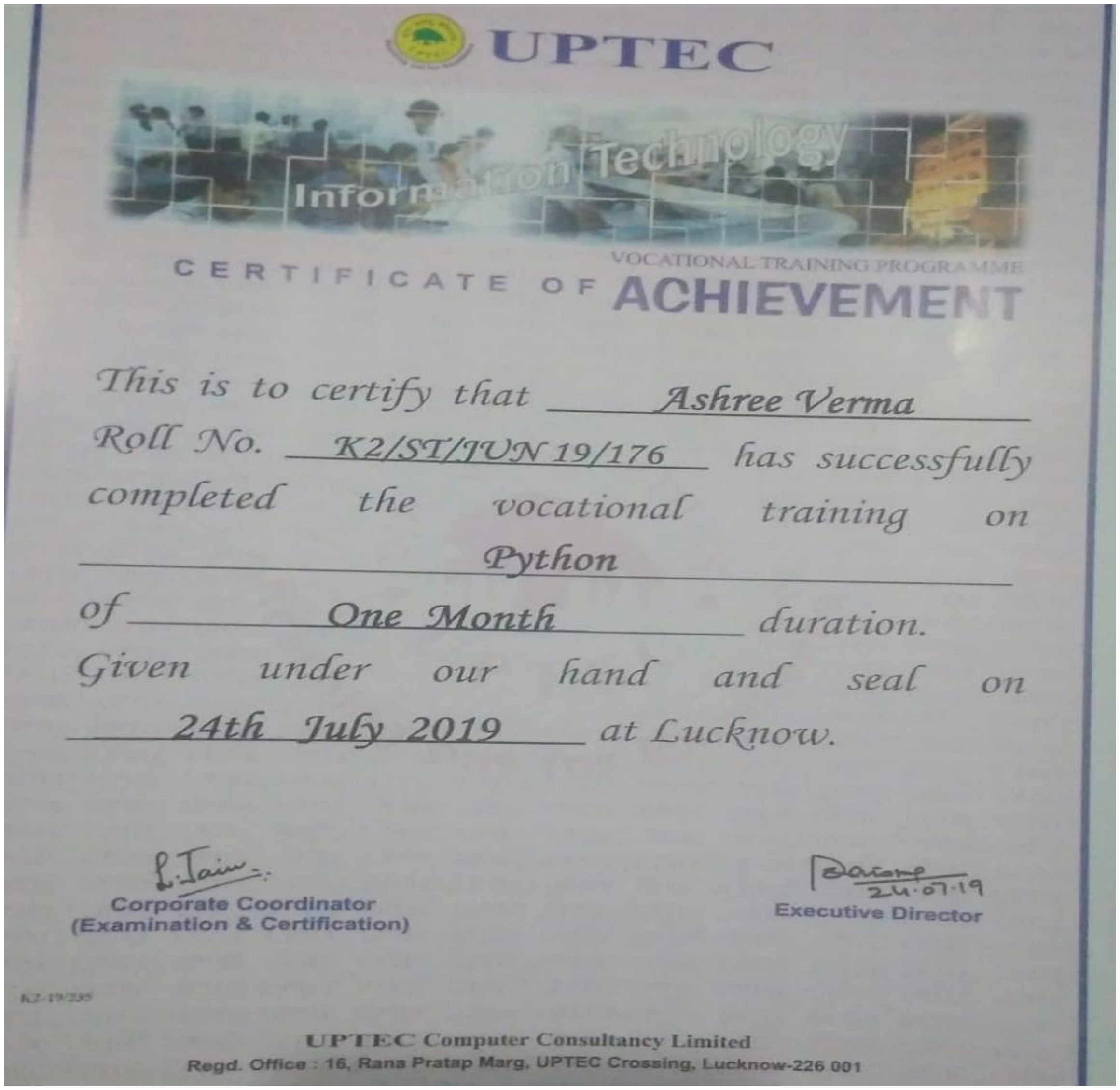
Mathura – 281406

ABSTRACT

Extracting speaker sentiment from natural audio streams such as YouTube is challenging. A number of factors contribute to the task difficulty, namely, Automatic Speech Recognition (ASR) of spontaneous speech, unknown background environments, variable source and channel characteristics, accents, diverse topics, *etc*.

Sentiment analysis or opinion mining is the field of study related to analyze opinions, sentiments, evaluations, attitudes, and emotions of users which they express on social media and other online resources. The revolution of social media sites has also attracted the users towards video sharing sites, such as YouTube. The online users express their opinions or sentiments on the videos they watch on such sites. This paper presents a brief survey of techniques to analyze opinions posted by users about a particular video.

CERTIFICATES



UPTEC

Information Technology

VOCATIONAL TRAINING PROGRAMME

CERTIFICATE OF ACHIEVEMENT

This is to certify that Ashree Verma
Roll No. K2/ST/JUN 19/176 has successfully completed the vocational training on
Python
of One Month duration.
Given under our hand and seal on
24th July 2019 at Lucknow.

L. Jain
Corporate Coordinator
(Examination & Certification)

[illegible]
24.07.19
Executive Director

K2-19/235

UPTEC Computer Consultancy Limited
Regd. Office : 16, Rana Pratap Marg, UPTEC Crossing, Lucknow-226 001

Department of computer
Engineering and Applications
GLA University, Mathura

17 km. Stone NH-2, Mathura-Delhi Road, P.O. – Chaumuhan,

Mathura – 281406

ACKNOWLEDGEMENT

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals. On the completion of this project, I would like to extend my sincere thanks to all of them. I am highly indebted to **Anniruddh Yadav sir, UPTEC** for his guidance. I wish to extend my sincere gratitude to **Prof. Anand Singh Jalal, Head of Department of Computer Engineering and Applications and faculty of CEA Department of GLA University** for his guidance, encouragement and give this opportunity and valuable suggestion which prove extremely useful and helpful in the completion of this report. I would also like to thank all those who directly or indirectly supported or helped me in completing my project in time. I would like to express my gratitude towards my parents and member of my college for their kind cooperation and encouragement which helped me in completion of this project. All of them have willingly helped me out with their abilities.

Thanks

ASHREE VERMA

Chapter – 1

Introduction

1.1 Overview

Social networking applications such as Twitter, Facebook, YouTube, etc. are popularly used to express one's sentiment and/or opinion on a variety of topics. A large number of these applications rely on text as the main medium of communication. However, websites such as YouTube use video/audio as the primary source of communicating information. For example, "unboxing" is a very popular theme on YouTube where users express their opinion and sentiment about products while unpacking and experiencing the product for the first time. Sentiment systems that can crawl and mine these information resources can assist in establishing the popular sentiment or the "word of mouth" on a large range of topics. Such information can be tremendously useful to businesses and consumers alike.

Text-based sentiment analysis has been well researched and numerous techniques that mine reviews for opinions have been developed. However, audio-based sentiment analysis remains under explored. Recently, we had shown that audio sentiment extraction with good accuracy is possible using a combination of NLP (natural language processing) and ASR(automatic speech recognition) techniques. Particularly, we had demonstrated the capability of automatically predicting the polarity of sentiment (positive or negative). First, audio was extracted from the YouTube video and then converted to text using the ASR system, and finally the text-based sentiment system predicted the sentiment polarity. The text-based sentiment system used parts-of-speech tagging technique to automatically extract text-features, which were then employed in a maximum entropy based classification system to predict sentiment polarity.

1.2 Motivation

When we do a YouTube search for videos, more often than not, it happens that the caption displayed at the bottom of video player doesn’t match appropriately with the content displayed. Our project aims to analyze the caption displayed and match it with the video content for the appropriateness, & in case of faulty or wrong caption, suggest a correct caption sequence.

It also aims to perform sentiment analysis on YouTube videos & perform categorization.

Chapter – 2
Software Requirement Analysis

2.1 Tools Used

Python IDLE - is an integrated development environment for Python, which has been bundled with the default implementation of the language.IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter.

Natural Language Toolkit - NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries.

NLTK has been called "a wonderful tool for teaching, and working in, computational linguistics using Python," and "an amazing library to play with natural language."

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The online version of the book has been updated for Python 3 and NLTK 3.

2.2 Problem

YouTube is overall a great platform, which explains how it has been popular for so long. However it does have its own share of flaws. Two of the biggest problems most users face are bad recommendation videos, and spam comments. It could help in better subtitles generation.

2.3 Modules and their Functionality

1. Interface of the application

In this module we use Tkinter:

The tkinter package is a thin object-oriented layer on top of Tcl/Tk. To use tkinter, you don’t need to write Tcl code, but you will need to consult the Tk documentation, and occasionally the Tcl documentation. tkinter is a set of wrappers that implement the Tk widgets as Python classes. In addition, the internal module _tkinterprovides a threadsafe mechanism which allows Python and Tcl to interact.

tkinter’s chief virtues are that it is fast, and that it usually comes bundled with Python. Although its standard documentation is weak, good material is available, which includes: references, tutorials, a book and others. tkinter is also famous for having an outdated look and feel, which has been vastly improved in Tk 8.5.

2. Video To Audio

In this module we use MoviePy Api.

MoviePy : MoviePy is an open source software originally written by Zulko and released under the MIT licence. It works on Windows, Mac, and Linux, with Python 2 or Python 3. The code is hosted on Github, where you can push improvements, report bugs and ask for help. There is also a MoviePy forum on Reddit and a mailing list on librelist**.**

MoviePy is a Python module for video editing, which can be used for basic operations (like cuts, concatenations, title insertions), video compositing (a.k.a. non-linear editing), video processing, or to create advanced effects. It can read and write the most common video formats, including GIF.

3. Audio To Text

In this module we use **Speech recognition Api.**

Speech recognition is the process of converting spoken words to text. Pythonsupports many speech recognition engines and APIs, including Google Speech Engine, Google Cloud Speech API, Microsoft Bing Voice Recognition and IBM Speech to Text.

The first component of speech recognition is, of course, speech. Speech must be converted from physical sound to an electrical signal with a microphone, and then to digital data with an analog-to-digital converter. Once digitized, several models can be used to transcribe the audio to text.

Most modern speech recognition systems rely on what is known as a Hidden Markov Model(HMM). This approach works on the assumption that a speech signal, when viewed on a short enough timescale (say, ten milliseconds), can be reasonably approximated as a stationary process—that is, a process in which statistical properties do not change over time.

Speech-to-Text API recognition

A Speech-to-Text API synchronous recognition request is the simplest method for performing recognition on speech audio data. Speech-to-Text can process up to 1 minute of speech audio data sent in a synchronous request. After Speech-to-Text processes and recognizes all of the audio, it returns a response.

A synchronous request is blocking. Speech-to-Text typically processes audio faster than realtime, processing 30 seconds of audio in 15 seconds on average. In cases of poor audio quality, your recognition request can take significantly longer.

4. Text To Polarity Analysis

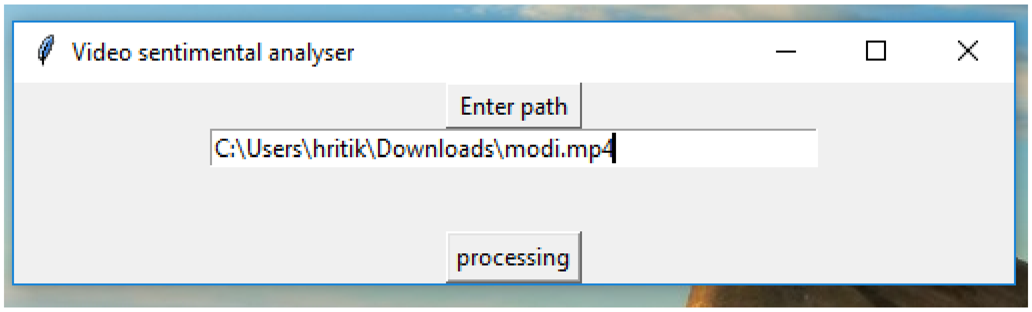
In this module we use **VADER Sentiment Analysis.**

VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive or negative.

VADER has been found to be quite successful when dealing with social media texts, NY Times editorials, movie reviews, and product reviews. This is because VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is.

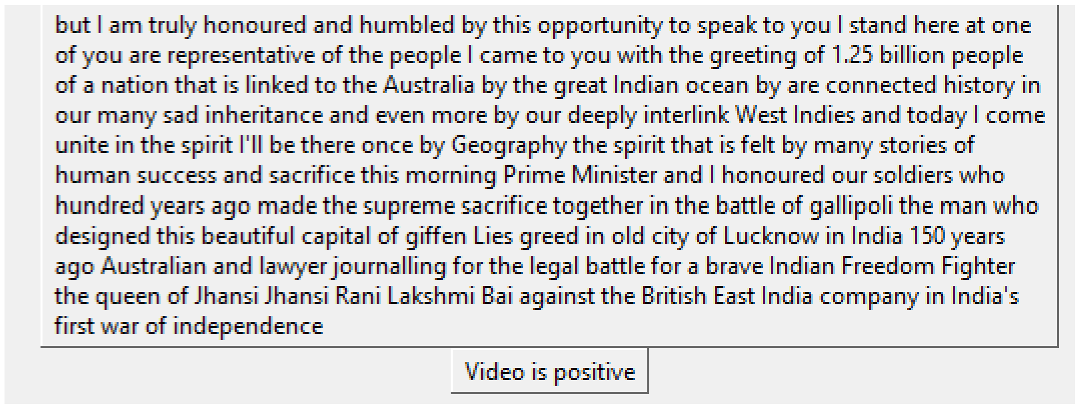
It is fully open-sourced under the MIT License. The developers of VADER have used Amazon’s Mechanical Turk to get most of their ratings, You can find complete details on their Github Page.

Input Format



C:\Users\hritik\Downloads\modi.mp4

Output Format



```
but I am truly honoured and humbled by this opportunity to speak to you I stand here at one
of you are representative of the people I came to you with the greeting of 1.25 billion people
of a nation that is linked to the Australia by the great Indian ocean by are connected history in
our many sad inheritance and even more by our deeply interlink West Indies and today I come
unite in the spirit I'll be there once by Geography the spirit that is felt by many stories of
human success and sacrifice this morning Prime Minister and I honoured our soldiers who
hundred years ago made the supreme sacrifice together in the battle of gallipoli the man who
designed this beautiful capital of giffen Lies greed in old city of Lucknow in India 150 years
ago Australian and lawyer journalling for the legal battle for a brave Indian Freedom Fighter
the queen of Jhansi Jhansi Rani Lakshmi Bai against the British East India company in India's
first war of independence
```

2.4 Requirements

1) **Hardware Requirements**
   - Minimum 4 GB RAM, and
   - Minimum i3 processor.
2) **Software Requirements**
   - Any windows based operating system,
   - Python,
   - NLTK, and
   - NLP Libraries.

Chapter – 3
Software Design

3.1 Sequence Diagram

Sequence Diagram is an interaction diagram that details how operations are carried out -- what messages are sent and when. Sequence diagrams are organized according to time.

The time progresses as you go down the page. The objects involved in the operation are listed from left to right according to when they take part in the message sequence.

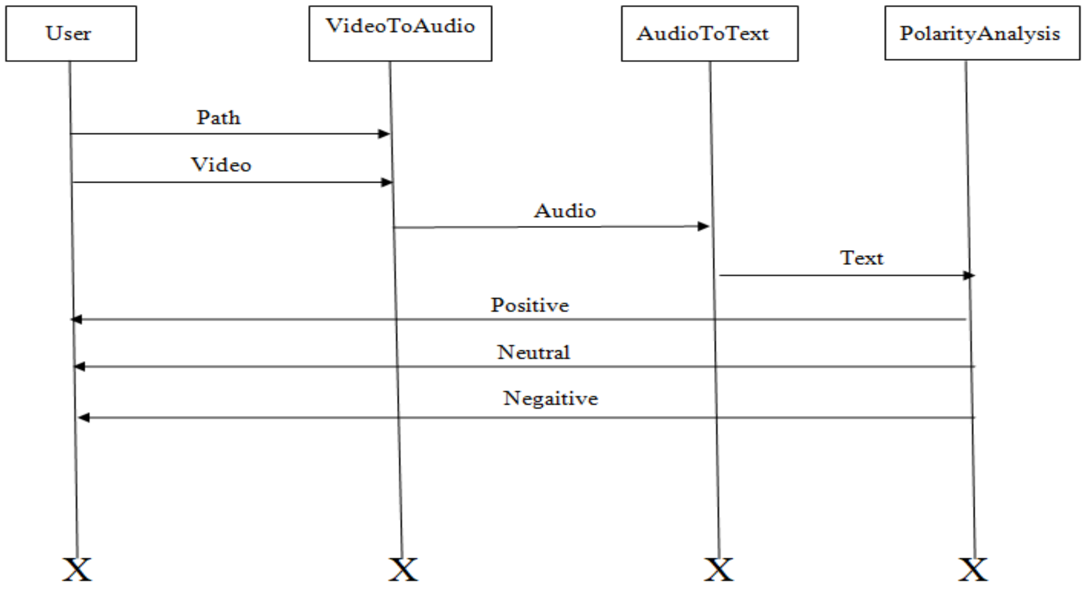


Fig. 3.1 Sequence Diagram

3.2 Data Flow Diagram

A Data Flow Diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination. Data flowcharts can range from simple even hand-drawn process overviews, to in-depth, multi-level DFDs that dig progressively deeper into how the data is handled. They can be used to analyze an existing system or model a new one. Like all the best diagrams and charts, a DFD can often visually “say” things that would be hard to explain in words, and they work for both technical and non technical audiences, from developer to CEO. That’s why DFDs is remains so popular after all these years. While they work well for dataflow software and systems, they are less applicable nowadays to visualizing interactive, real-time or database-oriented software or systems.

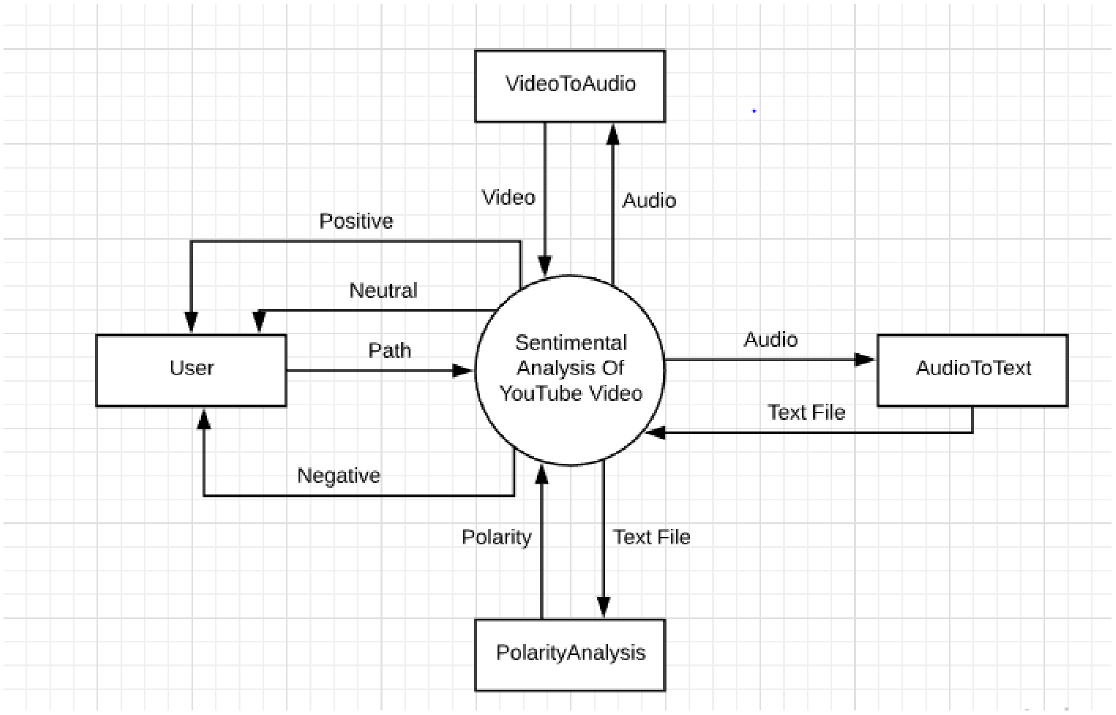


Fig. 3.2.1 Level 0 DFD

Chapter – 4

Libraries

4.1 Tkinter

Tk/Tcl has long been an integral part of Python. It provides a robust and platform independent windowing toolkit, that is available to Python programmers using the tkinter package, and its extension, the tkinter.tix and the tkinter.ttk modules.

The tkinter package is a thin object-oriented layer on top of Tcl/Tk. To use tkinter, you don’t need to write Tcl code, but you will need to consult the Tk documentation, and occasionally the Tcl documentation. tkinter is a set of wrappers that implement the Tk widgets as Python classes. In addition, the internal module _tkinterprovides a threadsafe mechanism which allows Python and Tcl to interact.

tkinter’s chief virtues are that it is fast, and that it usually comes bundled with Python. Although its standard documentation is weak, good material is available, which includes: references, tutorials, a book and others. tkinter is also famous for having an outdated look and feel, which has been vastly improved in Tk 8.5.

4.1.1 Tkinter package:

The tkinter package (“Tk interface”) is the standard Python interface to the Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, as well as on Windows systems. (Tk itself is not part of Python; it is maintained at ActiveState.)

Running python–m tkinter from the command line should open a window demonstrating a simple Tk interface, letting you know that tkinter is properly installed on your system, and also showing what version of Tcl/Tk is installed, so you can read the Tcl/Tk documentation specific to that version.

4.1.2. Tkinter Modules:

Most of the time, tkinter is all you really need, but a number of additional modules are available as well. The Tk interface is located in a binary module named _tkinter. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

In addition to the Tk interface module, tkinter includes a number of Python modules, tkinter.constantsbeing one of the most important. Importing tkinter will automatically import tkinter.constants, so, usually, to use Tkinter all you need is a simple import statement: import tkinter

4.1.3.Tkinter Modules

Most of the time, tkinter is all you really need, but a number of additional modules are available as well. The Tk interface is located in a binary module named _tkinter. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

In addition to the Tk interface module, tkinter includes a number of Python modules, tkinter.constantsbeing one of the most important. Importing tkinter will automatically import tkinter.constants, so, usually, to use Tkinter all you need is a simple import statement:

- import tkinter

Or, more often:

- from tkinter import *

class tkinter.**Tk**(*screenName=None*, *baseName=None*, *className='Tk'*, *useTk=1*)

The Tk class is instantiated without arguments. This creates a toplevel widget of Tk which usually is the main window of an application. Each instance has its own associated Tcl interpreter.

tkinter.**Tcl**(*screenName=None*, *baseName=None*, *className='Tk'*, *useTk=0*)

4.2. Speech recognition

Speech recognition is the process of converting spoken words to text. Pythonsupports many speech recognition engines and APIs, including Google Speech Engine, Google Cloud Speech API, Microsoft Bing Voice Recognition and IBM Speech to Text.

```
$ pip install SpeechRecognition
```

Once installed, you should verify the installation by opening an interpreter session and typing:

```
>>>

>>>importspeech_recognitionassr
>>>sr.__version__
'3.8.1'
```

Note: The version number you get might vary. Version 3.8.1 was the latest at the time of writing.

Go ahead and keep this session open. You'll start to work with it in just a bit.

SpeechRecognition *will* work out of the box if all you need to do is work with existing audio files. Specific use cases, however, require a few dependencies. Notably, the PyAudio package is needed for capturing microphone input.

You'll see which dependencies you need as you read further. For now, let's dive in and explore the basics of the package.

The Recognizer Class

All of the magic in SpeechRecognition happens with the `Recognizer` class.

The primary purpose of a `Recognizer` instance is, of course, to recognize speech. Each instance comes with a variety of settings and functionality for recognizing speech from an audio source.

Creating a `Recognizer` instance is easy. In your current interpreter session, just type:

```
>>>
>>>r=sr.Recognizer()
```

Each `Recognizer` instance has seven methods for recognizing speech from an audio source using various APIs. These are:

- recognize_bing(): Microsoft Bing Speech
- recognize_google(): Google Web Speech API
- recognize_google_cloud(): Google Cloud Speech - requires installation of the google-cloud-speech package
- recognize_houndify(): Houndify by SoundHound
- recognize_ibm(): IBM Speech to Text
- recognize_sphinx(): CMU Sphinx - requires installing PocketSphinx
- recognize_wit(): Wit.ai

Of the seven, only `recognize_sphinx()` works offline with the CMU Sphinx engine. The other six all require an internet connection.

A full discussion of the features and benefits of each API is beyond the scope of this tutorial. Since SpeechRecognition ships with a default API key for the Google Web Speech API, you can get started with it right away. For this reason, we'll use the Web Speech API in this guide. The other six APIs all require authentication with either an API key or a username/password combination. For more information, consult the SpeechRecognition docs.

Working With Audio Files

Before you continue, you'll need to download an audio file. The one I used to get started, "harvard.wav," can be found here. Make sure you save it to the same directory in which your Python interpreter session is running.

SpeechRecognition makes working with audio files easy thanks to its handy `AudioFile` class. This class can be initialized with the path to an audio file and provides a context manager interface for reading and working with the file's contents.

Supported File Types

Currently, SpeechRecognition supports the following file formats:

- WAV: must be in PCM/LPCM format
- AIFF
- AIFF-C
- FLAC: must be native FLAC format; OGG-FLAC is not supported

If you are working on x-86 based Linux, macOS or Windows, you should be able to work with FLAC files without a problem. On other platforms, you will need to install a FLAC encoder and ensure you have access to the `flac` command line tool. You can find more information here if this applies to you.

Using record() to Capture Data From a File

Type the following into your interpreter session to process the contents of the "harvard.wav" file:

```
>>>

>>>harvard=sr.AudioFile('harvard.wav')
>>>withharvardassource:
... audio=r.record(source)
...
```

The context manager opens the file and reads its contents, storing the data in an `AudioFile`instance called `source`. Then the `record()` method records the data from the entire file into an `AudioData` instance. You can confirm this by checking the type of `audio`:

```
>>>
```

```
>>>type(audio)
<class 'speech_recognition.AudioData'>
```

You can now invoke `recognize_google()` to attempt to recognize any speech in the audio. Depending on your internet connection speed, you may have to wait several seconds before seeing the result.

```
>>>
```

```
>>>r.recognize_google(audio)
'the stale smell of old beer lingers it takes heat
to bring out the odor a cold dip restores health and
zest a salt pickle taste fine with ham tacos al
Pastore are my favorite a zestful food is the hot
cross bun'
```

Capturing Segments With offset and duration

What if you only want to capture a portion of the speech in a file? The `record()` method accepts a `duration` keyword argument that stops the recording after a specified number of seconds.

For example, the following captures any speech in the first four seconds of the file:

```
>>>
```

```
>>>withharvardassource:
... audio=r.record(source,duration=4)
...
>>>r.recognize_google(audio)
'the stale smell of old beer lingers'
```

The `record()` method, when used inside a `with` block, always moves ahead in the file stream. This means that if you record once for four seconds and then record again for four seconds, the second time returns the four seconds of audio *after* the first four seconds.

```
>>>
```

```
>>>withharvardassource:
... audio1=r.record(source,duration=4)
... audio2=r.record(source,duration=4)
...
>>>r.recognize_google(audio1)
'the stale smell of old beer lingers'
>>>r.recognize_google(audio2)
'it takes heat to bring out the odor a cold dip'
```

To capture only the second phrase in the file, you could start with an offset of four seconds and record for, say, three seconds.

```
>>>
```

```
>>>withharvardassource:
... audio=r.record(source,offset=4,duration=3)
...
>>>recognizer.recognize_google(audio)
'it takes heat to bring out the odor'
```

The `offset` and `duration` keyword arguments are useful for segmenting an audio file *if* you have prior knowledge of the structure of the speech in the file. However, using them hastily can result in poor transcriptions. To see this effect, try the following in your interpreter:

```
>>>
```

```
>>>withharvardassource:
... audio=r.record(source,offset=4.7,duration=2.8)
...
```

```
>>>recognizer.recognize_google(audio)
'Mesquite to bring out the odor Aiko'
```

By starting the recording at 4.7 seconds, you miss the "it t" portion a the beginning of the phrase "it takes heat to bring out the odor," so the API only got "akes heat," which it matched to "Mesquite."

Similarly, at the end of the recording, you captured "a co," which is the beginning of the third phrase "a cold dip restores health and zest." This was matched to "Aiko" by the API.

There is another reason you may get inaccurate transcriptions. Noise! The above examples worked well because the audio file is reasonably clean. In the real world, unless you have the opportunity to process audio files beforehand, you can not expect the audio to be noise-free.

VADER Sentiment Analysis

VADER (**Valence Aware Dictionary and sEntiment Reasoner**) is a lexicon and rule-based sentiment analysis tool that is ***specifically attuned to sentiments expressed in social media***. VADER uses a combination of A sentiment lexicon is a list of lexical features (e.g., words) which are generally labelled according to their semantic orientation as either positive or negative.

VADER has been found to be quite successful when dealing with social media texts, NY Times editorials, movie reviews, and product reviews. This is because VADER not only tells **about** the Positivity and Negativity score but also tells us about **how positive or negative a sentiment is**.

It is fully open-sourced under the MIT License. The developers of VADER have used Amazon’s Mechanical Turk to get most of their ratings, You can find complete details on their Github Page

Working & Scoring

Let us test our first sentiment using VADER now. We will use the **polarity_scores()** method to obtain the polarity indices for the given sentence.

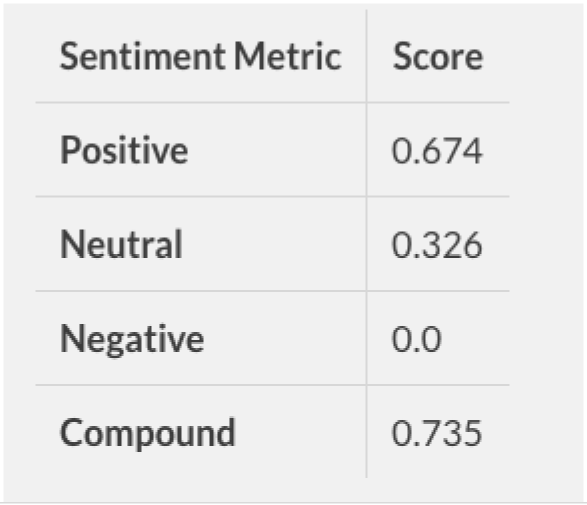
```
def sentiment_analyzer_scores(sentence):
   score = analyser.polarity_scores(sentence)
   print("{:-<40} {}".format(sentence, str(score)))
```

Let us check how VADER performs on a given review:

```
sentiment_analyzer_scores("The phone is super cool.")
```

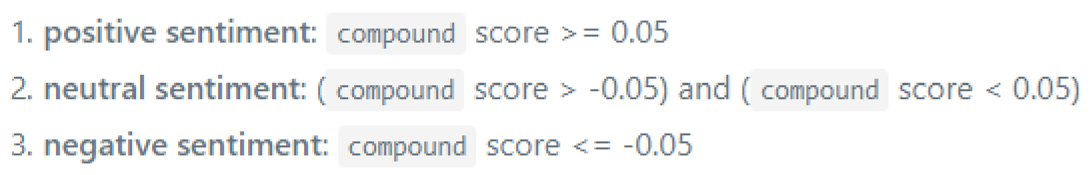
```
The phone is super cool----------------- {'neg': 0.0, 'neu': 0.326, 'pos': 0.674, 'compound': 0.7351}
```

Putting in a Tabular form:



Sentiment Metric	Score
Positive	0.674
Neutral	0.326
Negative	0.0
Compound	0.735

- The Positive, Negative and Neutral scores represent the proportion of text that falls in
- these categories. This means our sentence was rated as 67% Positive, 33% Neutral and 0% Negative. Hence all these should add up to 1.
- The Compound score is a metric that calculates the sum of all the lexicon ratings which have been normalized between -1(most extreme negative) and +1 (most extreme positive). In the case above, lexicon ratings for andsupercool are 2.9and respectively1.3. The compound score turns out to be 0.75 , denoting a very high positive sentiment.



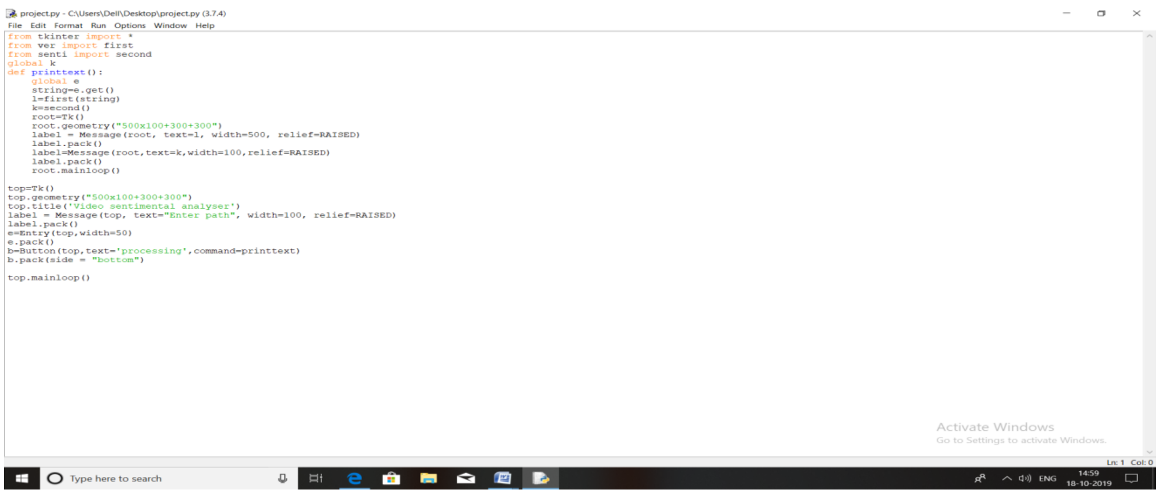
1. **positive sentiment**: `compound` score >= 0.05
2. **neutral sentiment**: (`compound` score > -0.05) and (`compound` score < 0.05)
3. **negative sentiment**: `compound` score <= -0.05

VADER analyses sentiments primarily based on certain key points

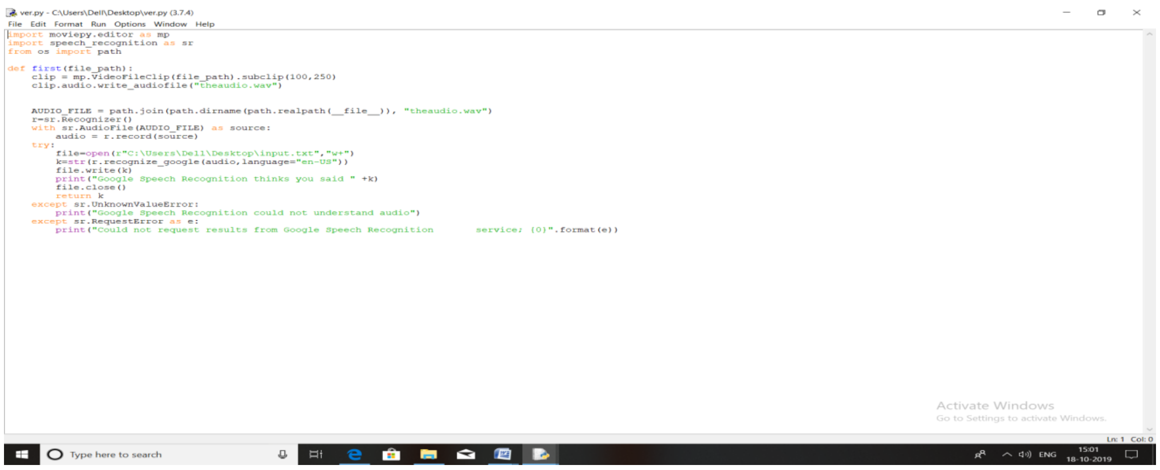
- **Punctuation: T**he use of an exclamation mark**(!)**, increases the magnitude of the intensity without modifying the semantic orientation.

Chapter – 5
Sample Code

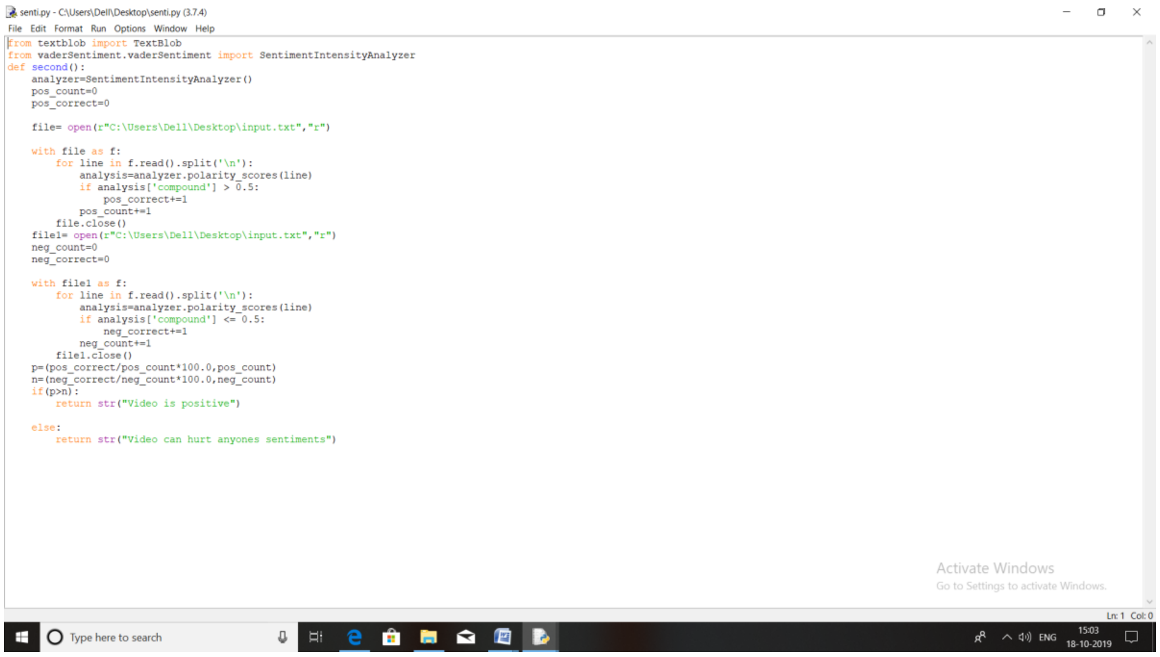
Sample code:


project.py - C:\Users\Dell\Desktop\project.py (3.7.4)
File Edit Format Run Options Window Help
Activate Windows
Go to Settings to activate Windows.
Ln: 1 Col: 0
Type here to search
14:59
18-10-2019

```python
from tkinter import *
from ver import first
from senti import second
global k
def printtext():
    global e
    string=e.get()
    l=first(string)
    k=second()
    root=Tk()
    root.geometry("500x100+300+300")
    label = Message(root, text=l, width=500, relief=RAISED)
    label.pack()
    label=Message(root,text=k,width=100,relief=RAISED)
    label.pack()
    root.mainloop()

top=Tk()
top.geometry("500x100+300+300")
top.title('Video sentimental analyser')
label = Message(top, text="Enter path", width=100, relief=RAISED)
label.pack()
e=Entry(top,width=50)
e.pack()
b=Button(top,text='processing',command=printtext)
b.pack(side = "bottom")

top.mainloop()
```


ver.py - C:\Users\Dell\Desktop\ver.py (3.7.4)
File Edit Format Run Options Window Help
Activate Windows
Go to Settings to activate Windows.
Ln: 1 Col: 0
Type here to search
15:01
18-10-2019

```python
import moviepy.editor as mp
import speech_recognition as sr
from os import path

def first(file_path):
    clip = mp.VideoFileClip(file_path).subclip(100,250)
    clip.audio.write_audiofile("theaudio.wav")


    AUDIO_FILE = path.join(path.dirname(path.realpath(__file__)), "theaudio.wav")
    r=sr.Recognizer()
    with sr.AudioFile(AUDIO_FILE) as source:
        audio = r.record(source)
    try:
        file=open(r"C:\Users\Dell\Desktop\input.txt","w+")
        k=str(r.recognize_google(audio,language="en-US"))
        file.write(k)
        print("Google Speech Recognition thinks you said " +k)
        file.close()
        return k
    except sr.UnknownValueError:
        print("Google Speech Recognition could not understand audio")
    except sr.RequestError as e:
        print("Could not request results from Google Speech Recognition       service; {0}".format(e))
```


senti.py - C:\Users\Dell\Desktop\senti.py (3.7.4)
File Edit Format Run Options Window Help
Activate Windows
Go to Settings to activate Windows.
Ln: 1 Col: 0
Type here to search
15:03
18-10-2019

```python
from textblob import TextBlob
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
def second():
    analyzer=SentimentIntensityAnalyzer()
    pos_count=0
    pos_correct=0

    file= open(r"C:\Users\Dell\Desktop\input.txt","r")

    with file as f:
        for line in f.read().split('\n'):
            analysis=analyzer.polarity_scores(line)
            if analysis['compound'] > 0.5:
                pos_correct+=1
            pos_count+=1
        file.close()
    file1= open(r"C:\Users\Dell\Desktop\input.txt","r")
    neg_count=0
    neg_correct=0

    with file1 as f:
        for line in f.read().split('\n'):
            analysis=analyzer.polarity_scores(line)
            if analysis['compound'] <= 0.5:
                neg_correct+=1
            neg_count+=1
        file1.close()
    p=(pos_correct/pos_count*100.0,pos_count)
    n=(neg_correct/neg_count*100.0,neg_count)
    if(p>n):
        return str("Video is positive")

    else:
        return str("Video can hurt anyones sentiments")
```

Chapter - 6

Reference/Bibliography

- https://www.quora.com/How-do-I-perform-sentiment-analysis-on-YouTube-videos
- https://www.github.com
- https://www.wikipedia.com
- https://ieeexplore.ieee.org/document/8120658/
- https://www.researchgate.net/publication/321408074_YouTube_video_by_sentiment_analysis
- https://stackoverflow.com/questions/44773553/what-i-can-use-to-process-video-sentiment-analysis-on-a-video-stream
- Natural Language Processing with python by Edward Loper, Steven Bird, Ewan Klein.