**A**

**Project III Report**

**On**

**Sentimental Analysis**



**Submitted for Partial Fulfillment of the Award of Bachelor of Technology (B.Tech) in CSE Kurukshetra University Kurukshetra**

**Submitted By:**

**Shivam Bembey (1219300)**

**Sudhanshu Bhardwaj (1219314)**

**Submitted to:**

**Er. Gaurav**

**(HOD, CSE)**

**Department of Computer Science and Engineering**

**Seth Jai Parkash Mukand Lal Institute of Engineering & Technology**

**Affiliated to Kurukshetra University Kurukshetra**

**Declaration**

We hereby certify that the work which is being presented in the Project III Report entitled, **“Report Title”** by us, **Shivam (1219300) , Sudhanshu(1219314)** in partial fulfilment of the requirements for the award of degree of **Bachelor of Technology** in **Computer Engineering** submitted in the **Department of Computer Science and Engineering at** JMIT **Radaur (Affiliated to Kurukshetra University Kurukshetra, Haryana (India))** is an authentic record of my/our own work carried out under the supervision of **Er. Alka** The matter presented in the report has not been submitted in any other Institute for the award of any degree.

**Shivam (1219300), Sudhanshu(1219314)**

This is to certify that the above statement made by the candidate is correct to the best of my/our knowledge.

**Er. Alka**

Professor Department of Computer Science and Engg. **JMIT Radaur**

**Countersigned By:** Er. Gaurav Sharma

The B.Tech Project III Viva-voce examination of **Shivam(1219300) , Sudhanshu(1219314)** has been held on ………………….. and is accepted.

**Er. Alka (Examiner)**

(Supervisor)

**Acknowledgement**

The writing of this project report has been assisted by the generous help of many people.we feel that we were very fortunate to receive assistance from them. I/we wish to express my/our sincere appreciation to them.

First and foremost, we are indebted to our principal supervisor, **Er. Alka (**Professor, Department of Computer Science and Engineering) of JMIT Radaur, who has been very supportive at every stage of my/our project completion. I/we wish to express my/our utmost gratitude to him/her for the invaluable advice and patience in reading, correcting and commenting on the drafts of this report and, more importantly, for his generosity which we have received throughout my/our project completion.

we would like to acknowledge and extended my/our heartfelt gratitude to **HOD Er. Gaurav Sharma** who helped and encouraged me/us throughout this journey.

I/we wish to express my/our thanks to all staff members of JMIT Radaur, who also helped me/us in conducting this study.

Finally, we are particularly indebted to our dearest parentsas without their generous assistance and love; this project could never have been completed.

**Shivam (1219300)**

**Sudhanshu (1219314)**

**TABLE OF CONTENTS**

Declaration………………...……………………………………………………………………….i Acknowledgement………………………………………………………………………………….ii Abstract ……………………………………………………………………………………………iii Table of Contents ………………………………………………………………………………….iv

|  |  |  |
| --- | --- | --- |
| **Chapter 1.** | I. | Problem Background and its Context |
|  | II. | Functionality (Core and Enhanced) |
|  | III. | Academic, Technical & Economic Feasibility |
| **Chapter 2.** | I. | Primary Research Techniques and Analysis |
|  | II. | Secondary Research (Existing System’s Study, SDLC, Proposed System’s |
|  |  | Architecture & Development Software’s Comparisons) |
| **Chapter 3.** | I. | Requirement Analysis (Use Cases/ SRS documentation etc.) |
|  | II. | System Design (DFD or UML (Activity, Class and Sequence Diagrams), |
|  |  | ERD/EERD) |

**Chapter 4.** System Implementation (Functional briefing with Module code/Pseudo-code)

**Chapter 5.** Future Scope & Limitations

**Chapter 6.** Conclusion

**Chapter 7.** Bibliography

**Chapter 8.** Appendix (Containing Project Meeting Log Sheets, Results and Screen Print-outs etc.)

**CHAPTER – 1**

**Introduction to Python**

Python is a [widely used](https://en.wikipedia.org/wiki/Measuring_programming_language_popularity) [high-level programming language](https://en.wikipedia.org/wiki/High-level_programming_language) for [general-purpose programming](https://en.wikipedia.org/wiki/General-purpose_programming_language), created by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) and first released in 1991. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [white space](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B) or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). The language provides constructs intended to enable writing clear programs on both a small and large scale.

Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management) and supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional programming](https://en.wikipedia.org/wiki/Functional_programming), and [procedural](https://en.wikipedia.org/wiki/Procedural_programming) styles. It has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system), allowing Python code to run on a wide variety of systems. [CPython](https://en.wikipedia.org/wiki/CPython" \o "CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation).

## [https://upload.wikimedia.org/wikipedia/commons/thumb/6/66/Guido_van_Rossum_OSCON_2006.jpg/220px-Guido_van_Rossum_OSCON_2006.jpg](https://en.wikipedia.org/wiki/File:Guido_van_Rossum_OSCON_2006.jpg)History

Python was conceived in the late 1980s, and its implementation began in December 1989 by [Guido van Rossum](https://en.wikipedia.org/wiki/Guido_van_Rossum) at [Centrum Wiskunde & Informatica](https://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica) (CWI) in the [Netherlands](https://en.wikipedia.org/wiki/Netherlands) as a successor to the [ABC language](https://en.wikipedia.org/wiki/ABC_(programming_language)) (itself inspired by [SETL](https://en.wikipedia.org/wiki/SETL)) capable of [exception handling](https://en.wikipedia.org/wiki/Exception_handling) and interfacing with the operating system [Amoeba](https://en.wikipedia.org/wiki/Amoeba_(operating_system)). Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, [Benevolent Dictator For Life](https://en.wikipedia.org/wiki/Benevolent_Dictator_For_Life) (BDFL).

About the origin of Python, Van Rossum wrote in 1996.

[**Guido van Rossum**](https://en.wikipedia.org/wiki/Guido_van_Rossum)**,**

**the creator of Python**

|  |
| --- |
|  |

Python 2.0 was released on 16 October 2000 and had many major new features, including a [cycle-detecting](https://en.wikipedia.org/wiki/Cycle_detection) [garbage collector](https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) and support for [Unicode](https://en.wikipedia.org/wiki/Unicode). With this release the development process was changed and became more transparent and community-backed.

Python 3.0 (that early in its development was commonly referred to as Python 3000 or py3k), a major, backward-incompatible release, was released on 3 December 2008[[33]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-3.0-release-33) after a long period of testing. Many of its major features have been [backported](https://en.wikipedia.org/wiki/Backporting" \o "Backporting) to the backwards-compatible Python 2.6.x and 2.7.x version series.

The End Of Life date (EOL, sunset date) for Python 2.7 was initially set at 2015, then postponed to 2020 out of concern that a large body of existing code cannot easily be forward-ported to Python 3. In January 2017, Google announced work on a Python 2.7 to [Go](https://en.wikipedia.org/wiki/Go_(programming_language)) [transcompiler](https://en.wikipedia.org/wiki/Transcompiler" \o "Transcompiler), which [The Register](https://en.wikipedia.org/wiki/The_Register) speculated was in response to Python 2.7's planned end-of-life but Google cited performance under concurrent workloads as their only motivation.

* **Syntax and semantics**

Python is intended to be a highly readable language. It is designed to have an uncluttered visual layout, often using English keywords where other languages use punctuation. Python does not use curly brackets to delimit blocks, and semicolons after statements are optional, in contrast to many other programming languages. Further, Python has fewer syntactic exceptions and special cases than [C](https://en.wikipedia.org/wiki/C_(programming_language)) or [Pascal](https://en.wikipedia.org/wiki/Pascal_(programming_language)).

* **Indentation**

Python uses [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [blocks](https://en.wikipedia.org/wiki/Block_(programming)) – rather than [curly braces](https://en.wikipedia.org/wiki/Curly_bracket_programming_language) or keywords. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block. This feature is also sometimes termed the [off-side rule](https://en.wikipedia.org/wiki/Off-side_rule).

* **Methods**

[Methods](https://en.wikipedia.org/wiki/Method_(programming)) on objects are [functions](https://en.wikipedia.org/wiki/Function_(programming)) attached to the object's class; the syntax instance.method(argument) is, for normal methods and functions, [syntactic sugar](https://en.wikipedia.org/wiki/Syntactic_sugar) for Class.method(instance, argument). Python methods have an explicit [self](https://en.wikipedia.org/wiki/This_(computer_programming)" \o "This (computer programming)) parameter to access [instance data](https://en.wikipedia.org/wiki/Instance_data), in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., [C++](https://en.wikipedia.org/wiki/C%2B%2B), [Java](https://en.wikipedia.org/wiki/Java_(programming_language)), [Objective-C](https://en.wikipedia.org/wiki/Objective-C), or [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language))).

* **Typing**

Python uses [duck typing](https://en.wikipedia.org/wiki/Duck_typing) and has typed objects but untyped variable names. Type constraints are not checked at [compile time](https://en.wikipedia.org/wiki/Compile_time); rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being [dynamically typed](https://en.wikipedia.org/wiki/Type_system#Dynamic_type_checking_and_runtime_type_information), Python is [strongly typed](https://en.wikipedia.org/wiki/Strongly_typed_programming_language), forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using [classes](https://en.wikipedia.org/wiki/Class_(computer_science)), which are most often used for [object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming). New [instances](https://en.wikipedia.org/wiki/Object_(computer_science)) of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes are instances of the [metaclass](https://en.wikipedia.org/wiki/Metaclass" \o "Metaclass) type (itself an instance of itself), allowing [metaprogramming](https://en.wikipedia.org/wiki/Metaprogramming" \o "Metaprogramming) and [reflection](https://en.wikipedia.org/wiki/Reflection_(computer_science)).

Before version 3.0, Python had two kinds of classes: old-style and new-style.[[68]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-classy-68) The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

* **Modules**

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference.

Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

### Example

The Python code for a module named aname normally resides in a file namedaname.py. Here is an example of a simple module, support.py −

def print\_func( par ):

print "Hello : ", par

return

* **The import Statement**

You can use any Python source file as a module by executing an import statement in some other Python source file. The import has the following syntax −

import module1[, module2[,... moduleN]

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module hello.py, you need to put the following command at the top of the script −

#!/usr/bin/python3

# Import module support

import support

# Now you can call defined function that module as follows

support.print\_func("Zara")

When the above code is executed, it produces the following result −

Hello : Zara

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening repeatedly, if multiple imports occur.

**CHAPTER – 2**

**GUI Programming**

Python provides various options for developing graphical user interfaces (GUIs). The most important features are listed below.

Tkinter − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.

wxPython − This is an open-source Python interface for wxWidgets GUI toolkit. You can find a complete tutorial on WxPython.

PyQt −This is also a Python interface for a popular cross-platform Qt GUI library. TutorialsPoint has a very good tutorial on PyQt .

JPython − JPython is a Python port for Java, which gives Python scripts seamless access to the Java class libraries on the local machine .

There are many other interfaces available, which you can find them on the net.

* **Tkinter Programming**

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

Import the Tkinter module.

Create the GUI application main window.

Add one or more of the above-mentioned widgets to the GUI application.

Enter the main event loop to take action against each event triggered by the user.

Example

#!/usr/bin/python3

importtkinter # note that module name has changed from Tkinter in Python 2 to tkinter in Python 3

top = tkinter.Tk()

# Code to add widgets will go here...

top.mainloop()

This would create a following window –



* **Tkinter Widgets**

Tkinter provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets.

* **Tkinter Button**

The Button widget is used to add buttons in a Python application. These buttons can display text or images that convey the purpose of the buttons. You can attach a function or a method to a button which is called automatically when you click the button.

**Syntax**

Here is the simple syntax to create this widget −

w = Button ( master, option = value, ... )

**Parameters**

master − This represents the parent window.

options − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

Example

Try the following example yourself −

# !/usr/bin/python3

fromtkinter import \*

fromtkinter import messagebox

top = Tk()

top.geometry("100x100")

defhelloCallBack():

msg = messagebox.showinfo( "Hello Python", "Hello World")

B = Button(top, text = "Hello", command = helloCallBack)

B.place(x = 50,y = 50)

top.mainloop()

**Result**

When the above code is executed, it produces the following result –

###### TK Button

* **TkinterCheckbutton**

The Checkbutton widget is used to display a number of options to a user as toggle buttons. The user can then select one or more options by clicking the button corresponding to each option.

You can also display images in place of text.

**Syntax**

Here is the simple syntax to create this widget −

w = Checkbutton( master, option, ... )

**Parameters**

master − This represents the parent window.

options − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

Example

Try the following example yourself −

# !/usr/bin/python3

fromtkinter import \*

importtkinter

top = Tk()

CheckVar1 = IntVar()

CheckVar2 = IntVar()

C1 = Checkbutton(top, text = "Music", variable = CheckVar1, \

onvalue = 1, offvalue = 0, height=5, \

width = 20, )

C2 = Checkbutton(top, text = "Video", variable = CheckVar2, \

onvalue = 1, offvalue = 0, height=5, \

width = 20)

C1.pack()

C2.pack()

top.mainloop()

**Result**

When the above code is executed, it produces the following result –



* **Tkinter Entry**

The Entry widget is used to accept single-line text strings from a user.

If you want to display multiple lines of text that can be edited, then you should use the Text widget.

If you want to display one or more lines of text that cannot be modified by the user, then you should use the Label widget.

**Syntax**

Here is the simple syntax to create this widget −

w = Entry( master, option, ... )

**Parameters**

master − This represents the parent window.

options − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

Example

Try the following example yourself –

# !/usr/bin/python3

fromtkinter import \*

top = Tk()

L1 = Label(top, text = "User Name")

L1.pack( side = LEFT)

E1 = Entry(top, bd = 5)

E1.pack(side = RIGHT)

top.mainloop()

**Result**

When the above code is executed, it produces the following result –

TK Entry

.

* **Tkinter Label**

This widget implements a display box where you can place text or images. The text displayed by this widget can be updated at any time you want.

It is also possible to underline part of the text (like to identify a keyboard shortcut) and span the text across multiple lines.

**Syntax**

Here is the simple syntax to create this widget −

w = Label ( master, option, ... )

Parameters

master − This represents the parent window.

options − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

Example

Try the following example yourself −

# !/usr/bin/python3

fromtkinter import \*

root = Tk()

var = StringVar()

label = Label( root, textvariable = var, relief = RAISED )

var.set("Hey!? How are you doing?")

label.pack()

root.mainloop()

**Result**

When the above code is executed, it produces the following result −

### TK Label

* **Tkinter Frame**

The Frame widget is very important for the process of grouping and organizing other widgets in a somehow friendly way. It works like a container, which is responsible for arranging the position of other widgets.

It uses rectangular areas in the screen to organize the layout and to provide padding of these widgets. A frame can also be used as a foundation class to implement complex widgets.

**Syntax**

Here is the simple syntax to create this widget −

w = Frame ( master, option, ... )

Parameters

master − This represents the parent window.

options − Here is the list of most commonly used options for this widget. These options can be used as key-value pairs separated by commas.

Example

Try the following example yourself −

# !/usr/bin/python3

fromtkinter import \*

root = Tk()

frame = Frame(root)

frame.pack()

bottomframe = Frame(root)

bottomframe.pack( side = BOTTOM )

redbutton = Button(frame, text = "Red", fg = "red")

redbutton.pack( side = LEFT)

greenbutton = Button(frame, text = "Brown", fg="brown")

greenbutton.pack( side = LEFT )

bluebutton = Button(frame, text = "Blue", fg = "blue")

bluebutton.pack( side = LEFT )

blackbutton = Button(bottomframe, text = "Black", fg = "black")

blackbutton.pack( side = BOTTOM)

root.mainloop()

**Result**

When the above code is executed, it produces the following result –



* **TkintertkMessageBox**

The tkMessageBox module is used to display message boxes in your applications. This module provides a number of functions that you can use to display an appropriate message.

Some of these functions are showinfo, showwarning, showerror, askquestion, askokcancel, askyesno, and askretryignore.

**Syntax**

Here is the simple syntax to create this widget –

tkMessageBox.FunctionName (title, message [, options])

**Parameters**

FunctionName − This is the name of the appropriate message box function.

title − This is the text to be displayed in the title bar of a message box.

message − This is the text to be displayed as a message.

options − options are alternative choices that you may use to tailor a standard message box. Some of the options that you can use are default and parent. The default option is used to specify the default button, such as ABORT, RETRY, or IGNORE in the message box. The parent option is used to specify the window on top of which the message box is to be displayed.

You could use one of the following functions with dialogue box −

showinfo()

showwarning()

showerror ()

askquestion()

askokcancel()

askyesno ()

askretrycancel ()

Example

Try the following example yourself −

# !/usr/bin/python3

fromtkinter import \*

fromtkinter import messagebox

top = Tk()

top.geometry("100x100")

def hello():

messagebox.showinfo("Say Hello", "Hello World")

B1 = Button(top, text = "Say Hello", command = hello)

B1.place(x = 35,y = 50)

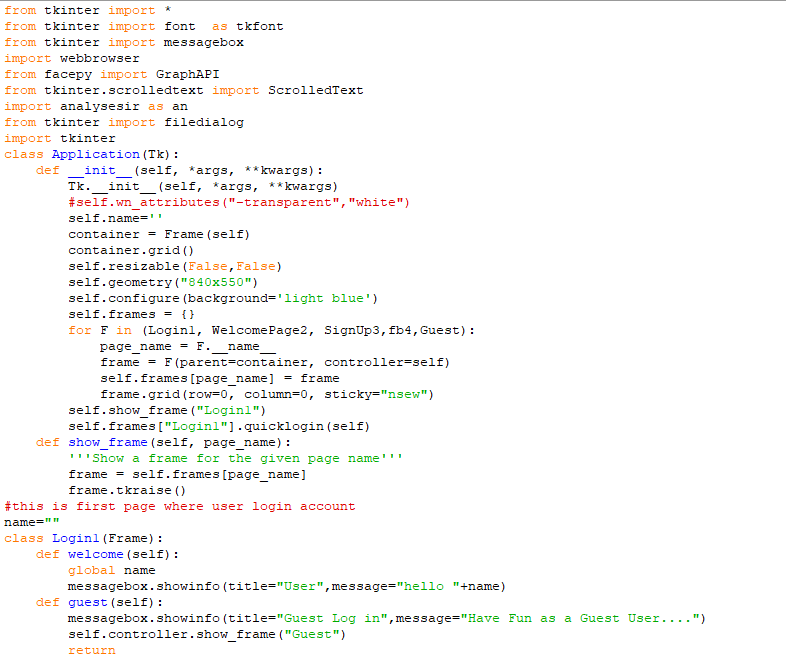
top.mainloop()

**CHAPTER – 3**

**(System Implementation (Functional briefing with Module code/Pseudo-code)**

**Below is source code for Sentimental Analysis**

**This is first page where user login account**

****

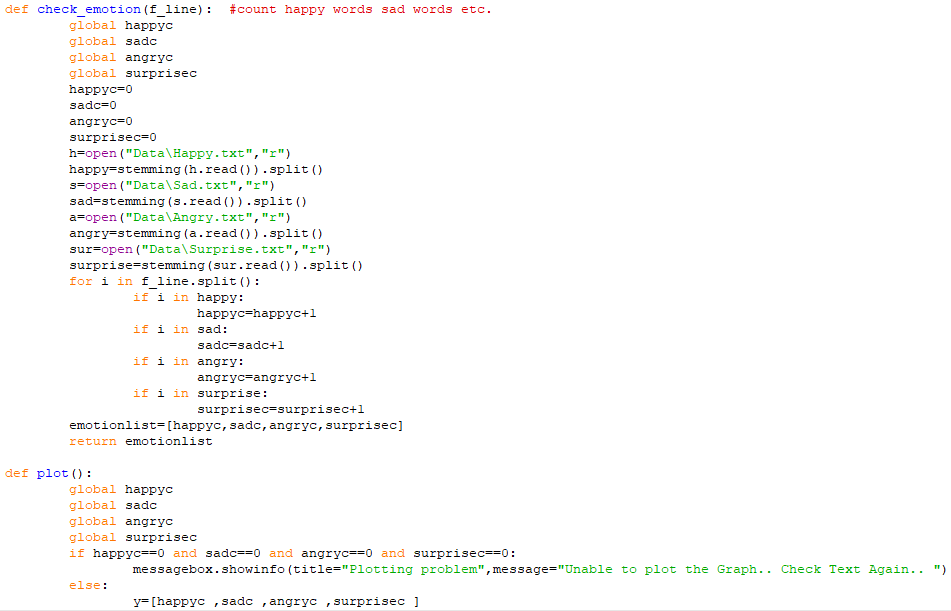
****

**Signing Up**

****

**Key Steps:- Tokenisation, Stop Word Removal, Removing Suffix and Counting Words**

****

****

# ****Text Normalization****

Normalizing the text means converting it to a more convenient, standard form before performing turning it to features for higher level modeling. Think of this step as converting human readable language into a form that is machine readable.

The standard framework to normalize the text includes:

1. Tokenization
2. Stop Words Removal
3. Morphological Normalization
4. Collocation

Data preprocessing consists of a number of steps, any number of which may or not apply to a given task. More generally, in this article we’ll discuss some predetermined body of text, and perform some basic transformative analysis that can be used for performing further, more meaningful natural language processing [1]

# ****Tokenization****

Given a character sequence and a defined document unit (blurb of texts), tokenization is the task of chopping it up into pieces, called **tokens**, perhaps at the same time throwing away certain characters/words, such as punctuation [2]. Ordinarily, there are two types of tokenization:

1. **Word Tokenization:**  Used to separate words via unique space character. Depending on the application, word tokenization may also tokenize multi-word expressions like New York. This is often times is closely tied to a process called Named Entity Recognition. Later in this tutorial, we will look at Collocation (Phrase) Modeling that helps address part of this challenge
2. **Sentence Tokenization/Segmentation:** Along with word tokenization, sentence segmentation is a crucial step in text processing. This is usually performed based on punctuations such as “.”, “?”, “!” as they tend to mark the sentence boundaries

# Stop Words Removal

Often, there are a few ubiquitous words which would appear to be of little value in helping the purpose of analysis but increases the dimensionality of feature set, are excluded from the vocabulary entirely as the part of stop words removal process. There are two considerations usually that motivate this removal.

1. **Irrelevance:** Allows one to analyze only on content-bearing words. Stopwords, also called empty words because they generally do not bear much meaning, introduce noise in the analysis/modeling process
2. **Dimension:** Removing the stopwords also allows one to reduce the tokens in documents significantly, and thereby decreasing feature dimension.

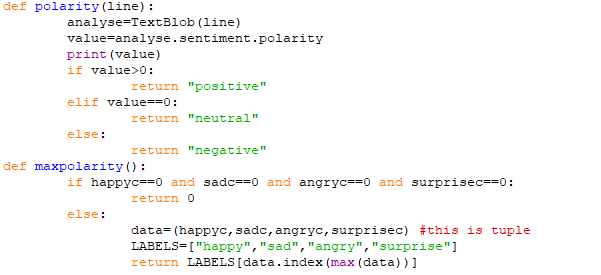
# Morphological Normalization

Morphology, in general, is the study of the way words are built up from smaller meaning-bearing units, morphomes. For example, dogs consists of two morphemes: **dog** and **s**

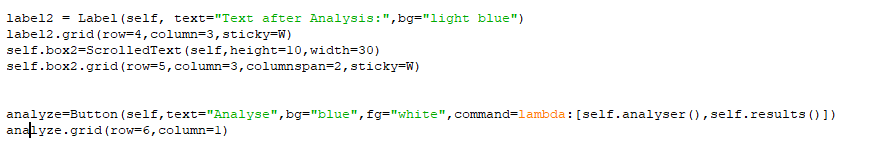
Two commonly used techniques for text normalization are:

1. **Stemming:** The procedure aims to identify the stem of a word and use it in lieu of the word itself. The most popular algorithm for stemming English, and one that has repeatedly been shown to be empirically very effective, is Porter’s algorithm. The entire algorithm is too long and intricate to present here.
2. **Lemmatization:**  This process refers to doing things correctly with the use of vocabulary and morphological analysis of words, typically aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the **lemma**.

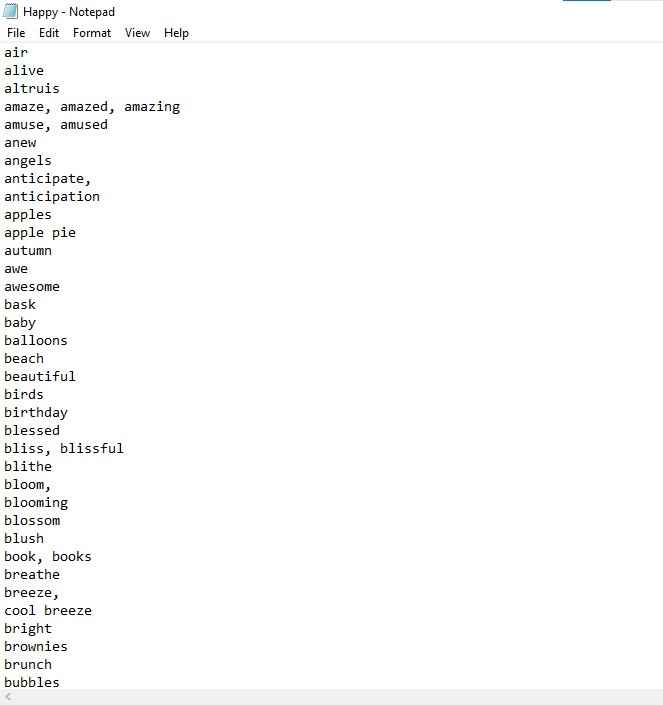
**Finding Polarity of Text:**

****

**Text after Analysis:**

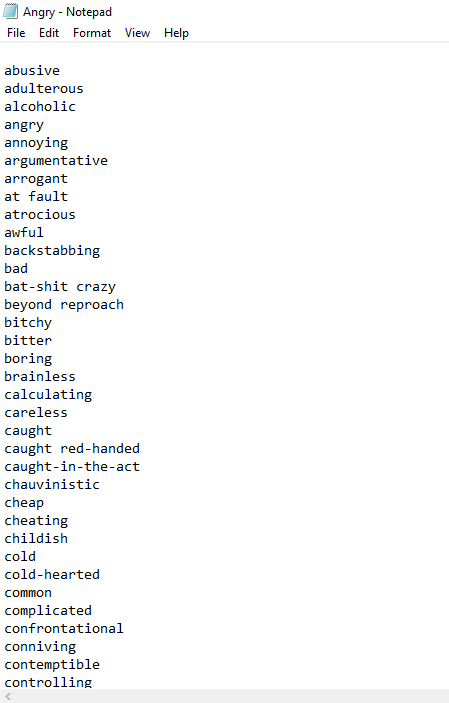
****

**Collection of words relatively similar or synonymous to Happy:**

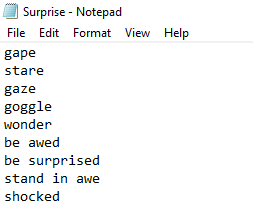
****

**Collection of words relatively similar or synonymous to Sad:**

****

**Collection of words relatively similar or synonymous to Angry:**

**Collection of words relatively similar or synonymous to Surprise:**

****

**Chapter -4**

**Project Work**

* **AIM & Objective:**

The Aim of My Project is to Analyse the Emotions of the User by the medium of textual sentences and their facebook Posts entered.

What we will Do?

Firstly we will get the user to enter the some text related to emotions or anything. Then we will use built in libraries of Python to analyse text and then check the mood of the User. In this project we will also analyse the user emotions by the way of getting user posts posted recently on the facebook.

* **Software Requirements:**

Python 3

Working Internet connection

Pip software for installation of packages

* **Python 3:**

Python 3.0 was released in 2008. Although this version is supposed to be backward incompatibles, later on many of its important features have been backported to be compatible with version 2.7.This tutorial gives enough understanding on Python 3 version programming language.

* **Pip:**

pip is a [package management system](https://en.wikipedia.org/wiki/Package_manager) used to install and manage [software packages](https://en.wikipedia.org/wiki/Package_(package_management_system)) written in [Python](https://en.wikipedia.org/wiki/Python_(programming_language)). Many packages can be found in the [Python Package Index](https://en.wikipedia.org/wiki/Python_Package_Index) (PyPI).

Python 2.7.9 and later (on the python2 series), and Python 3.4 and later include pip (pip3 for Python 3) by default.

pip is a [recursive acronym](https://en.wikipedia.org/wiki/Recursive_acronym) that can stand for either "Pip Installs Packages" or "Pip Installs Python".

One major advantage of pip is the ease of its [command-line interface](https://en.wikipedia.org/wiki/Command-line_interface), which makes installing Python software packages as easy as issuing one command:

pip install some-package-name

Users can also easily remove the package:

pip uninstall some-package-name

* **Future of Sentiment Analysis**

Sentiment Analysis has been more than just a social analytic tool. It’s been an interesting field of study. But it is a field that is still being studied, although not at great lengths due to the intricacy of this analysis. That is this field has functions that are too complicated for machines to understand. The ability to understand sarcasm, hyperbole, positive feelings, or negative feelings has been difficult, for machines that lack feelings. Algorithms have not been able to predict with more than 60% accuracy the feelings portrayed by people. Yet with so many limitations this is one field which is growing at great pace within many industries. Companies want to accommodate the sentiment analysis tools into areas of customer feedback, marketing, CRM, and ecommerce.

Sentiment analysis methods till now have been used to detect the polarity in the thoughts and opinions of all the users that access social media. Researchers and Businesses are very interested to understand the thoughts of people and how they respond to everything happening around them. Companies use this to evaluate their advertisement campaigns and to improve their products.

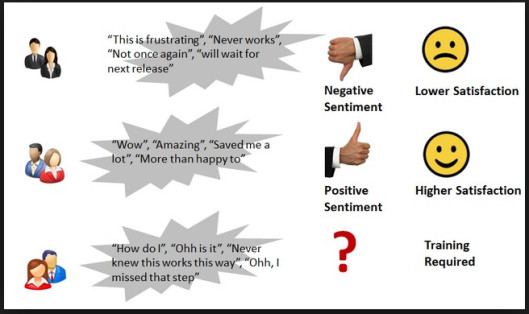
There is too much potential in machine learning, overtaking some of the manual labor of some lexicon based tasks that are labor intensive. For example, lexicon sentiment creation is labor intensive and there are already unsupervised methods to create them. This is where machine learning will play a crucial role. Such algorithms will also have to understand and analyze natural text concept-wise and context-wise. Time will also be a crucial element looking at the amount of data that is being generated on the Web today. Collecting opinions on the web will still requires processing that can filter out un-opinionated user-generated content and also to test the trustworthiness of the opinion and its source.

There is a lot of scope in analyzing the video and images on the web. Now a days, with the advent of Facebook, Instagram and Video vines people are expressing their thoughts with pictures and videos along with text. Sentiment analysis will have to pace up with this change. Tools which are helping companies to change strategies based on Facebook and Twitter will also have to accommodate the number of likes and re-tweets that the thought is generating on the Social media. People follow and unfollow people and comments on Social Media but never comment so there is scope in analyzing these aspects of the Web as well.

The use of punctuation is an obstacle in Sentiment Analysis which is under research as well. Sentiment Analysis has started helping us to predict events just like in the case of Obama vs Romney but is still naïve in most cases. A sentiment analysis tool Teamview had predicted the winner of the show X factor but eventually that person came second. So improvements on the analysis is one scope which is under way by many tools available on the web.

As new text types appear on the Social Web, the techniques to pre-process, as well as to tackle their informal style must be adapted, so as to obtain acceptable levels of performance of the sentiment analysis systems. The field will have to combine with effective computing, psychology and neuroscience to converge on a unified approach to understanding the sentiments better.

Many tools and algorithms rely on the polarity of the words and the scoring is dependent on this polarity. This means that accuracy drops since the semantics of the complete sentence is lost. The semantics of the sentence makes it difficult to measure the polarity of the sentences on individual words. For eg. “This car is anything but useful”. The word useful can make this sentence positive but eventually this is a negative sentence overall. There are a few limitation to sentiment analysis which are hampering the progress of the accuracy of the models.

[](https://datasciencecmu.files.wordpress.com/2014/04/ads31.jpg)

The positive or negative word might mean completely opposite depending upon the context used in the sentence. For example “My car is very good at using up the petrol at a faster rate.” Then sometimes the sentence ambiguity can be a problem since some positive or negative words might mean nothing in perspective of the sentence and sometimes words with no individual meaning express a lot of sentiment in the sentence. Sarcasm is the biggest challenge that sentiment analysis faces. Machine or algorithms with no emotion will find it extremely difficult to differentiate when users are commenting sarcastically.

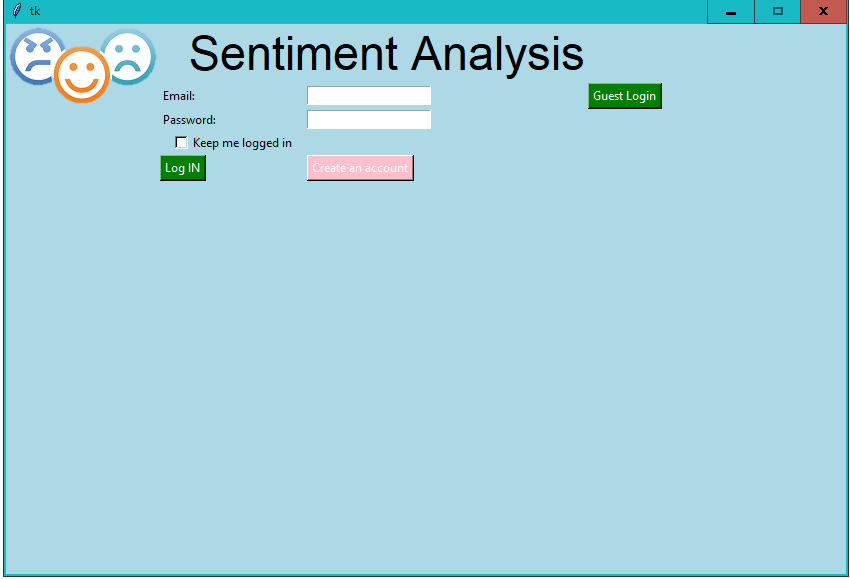
The language used throughout social media is different. Financial industry have their own language which means completely differs from Entertainment industry. This makes it hard for nay tool to predict the emotion or semantic of the sentence. People also use a lot of slang language and hashtags which makes the accuracy of the algorithms lower. It is difficult for the tool to even understand who the object of the sentence is. For example “I feel the browser is working fine but my friend hates working on it”.

**Sustenance**

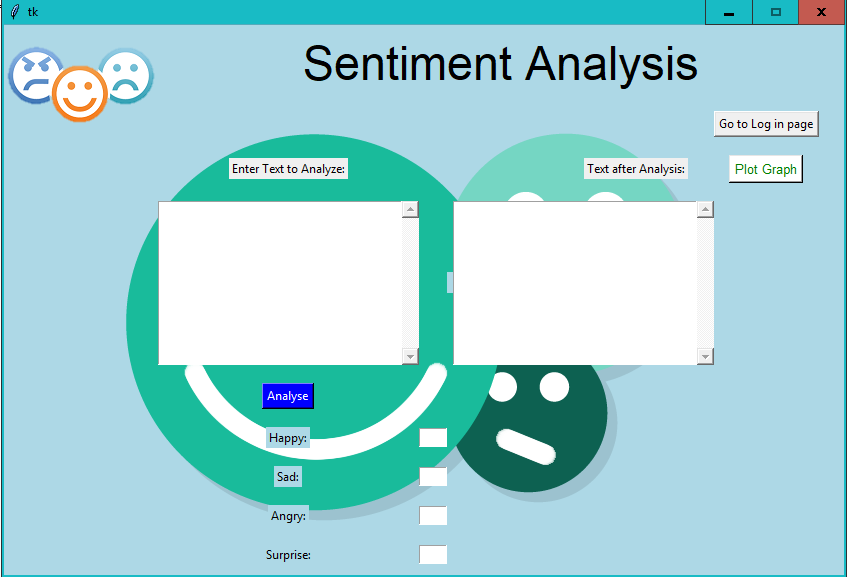
Sentiment analysis is not all that smooth after all. There are several issues related to Sentiment analysis that could lead to the loss of popularity of the technique.

* Opinion spam: Sentiment analysis can be used by competitors to portray negative image of a company. Once sentiment analysis gains popularity as a metric to gauge performance and brand image of a company, such mal-practices may become very common which will lead to decreased popularity of Sentiment Analysis.
* Result measure: The outputs of Sentiment analysis are useful as a reactive measure. It cannot be used to predict the performance of a company or other metrics. In some cases, Sentiment analysis can be redundant and can be only a reporting measure after the damage has been done.
* Lack of complete information. Biased results based on the sources: The sources of extracting information can be a major roadblock in sentiment analysis. Analysis of a scenario on incomplete information can lead to skewed results. Sources like Twitter, Facebook can be mined to get complete information.

**Screenshots of Project**

****

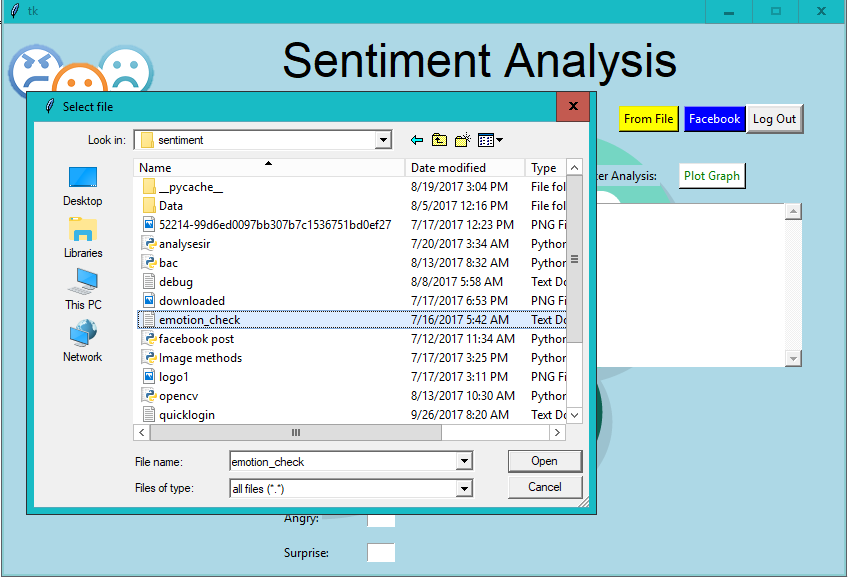
**Login Page**

****

**Guest Login Page**

### C:\Users\patiala\Desktop\project images\congrats.PNG

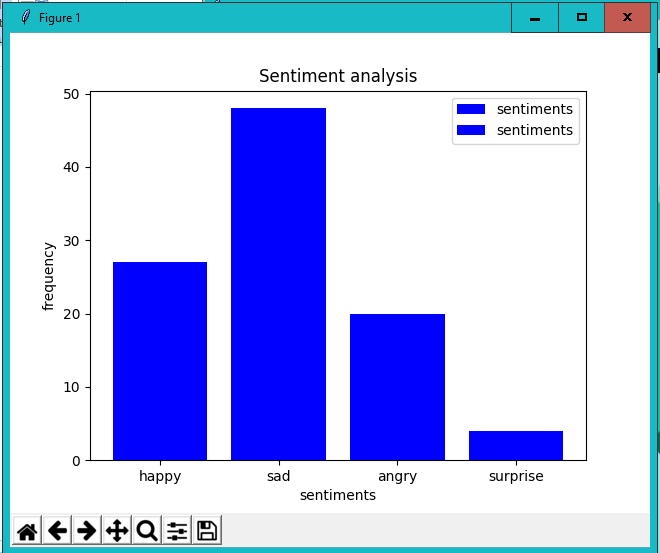
**Confirmation of Successful Login**

****

**Data Fetch From File Manually**

### C:\Users\patiala\Desktop\project images\plot1.PNG

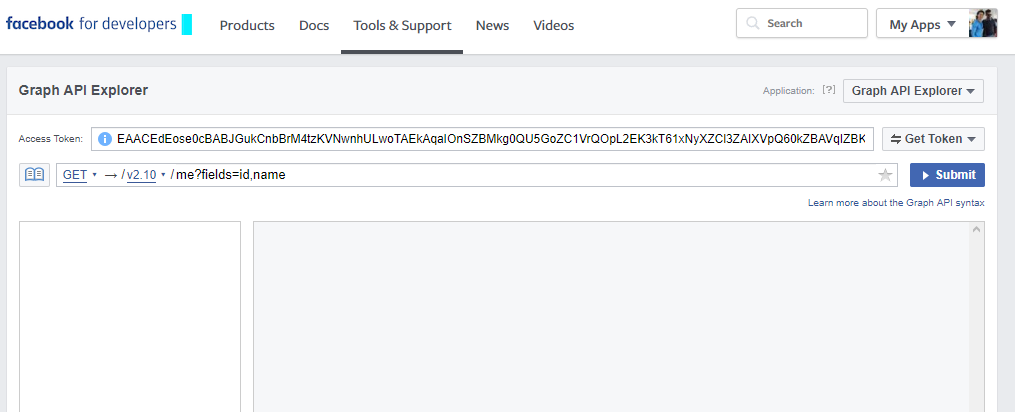
**Emotion Analysis after inserting the data**

****

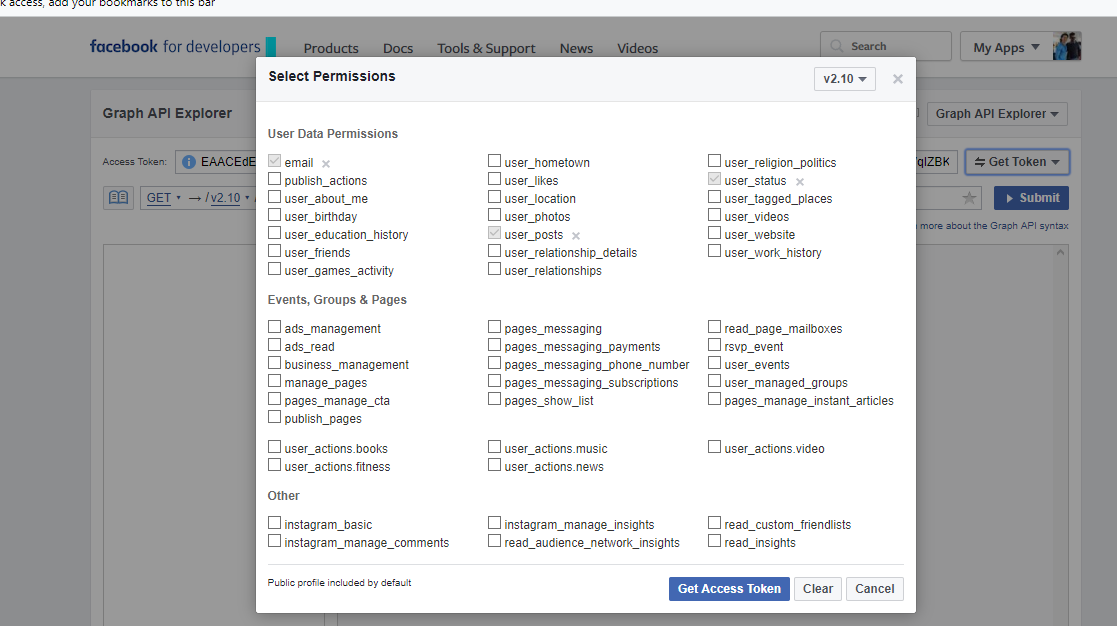
**Graph Plotting after results**

### C:\Users\patiala\Desktop\project images\facebook.PNG

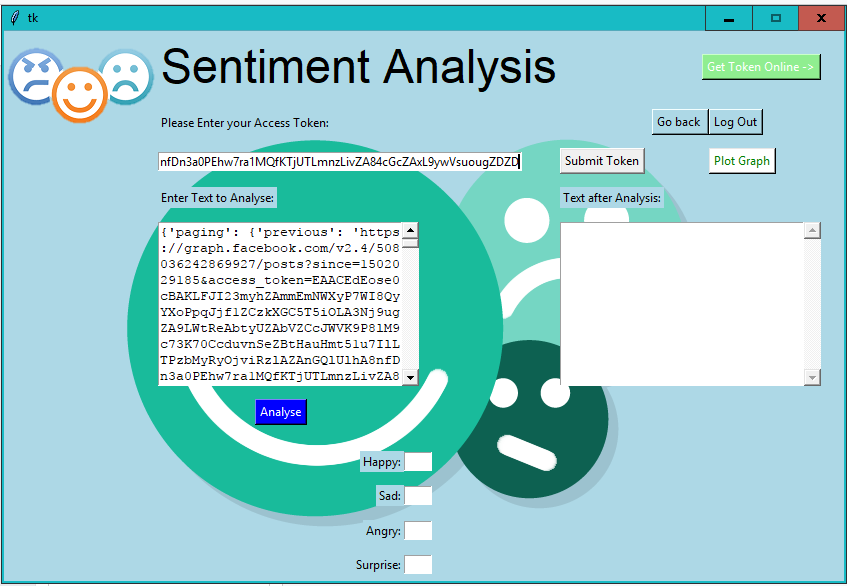
**Page Getting Facebook Posts**

****

**Facebook API Permission**



**User Access Token**

****

**Recent Facebook Posts after submitting Token**

**\**

**Chapter - 5**

**Results and discussion**

Online content is published so fast and furious these days that [no one can read it all](http://www.youtube.com/watch?v=sP4NMoJcFd4). Ears are burning at corporate brands, too, because much of that Web content is talking about them. What are people saying? One part of answering that question is to ask how people seem to be feeling about the things they are talking about. Scalable, automated, accurate, sophisticated sentiment analysis is a much sought-after technology that almost no one has really nailed yet.

[Jennifer Zeszut](https://twitter.com/#!/jenniferland) was the founder and CEO of [Scout Labs](http://scoutlabs.com/), a social media monitoring service [acquired by social CRM company Lithium](http://blog.louisgray.com/2010/05/scout-labs-acquired-by-lithium-to.html) in 2010. ScoutLabs does sentiment analysis, among other things, and Zeszut spoke at the [O’Reilly Strata Summit on Big Data](http://strataconf.com/) this week about the things her company has done that she believes point toward the future of this red hot tech trend.

Zeszut worked previously at eBay and Avenue A I Razorfish. She recently [addressed the White House](http://www.wajam.com/shorturl.php?se=google&url=http%3A%2F%2Fwww.montereyherald.com%2Fci_17250232%26source%3Dtweetmeme&id_user=37629) on how to best support tech startups.

Here’s a summary of her advice about sentiment analysis, for companies that build it and customers that buy it. My take-away here? It’s not just technology, it’s also a lot about muscle and brains.

Harness the wisdom of crowds – scale and let your algorithm learn. Zeszut says that once ScoutLabs got enough scale, across hundreds of companies with hundreds of users at many of them, it was common for one user to manually reclassify the sentiment on an item that was relevant to other users as well. Zeszut says that making sure her algorithm learned from that was key.

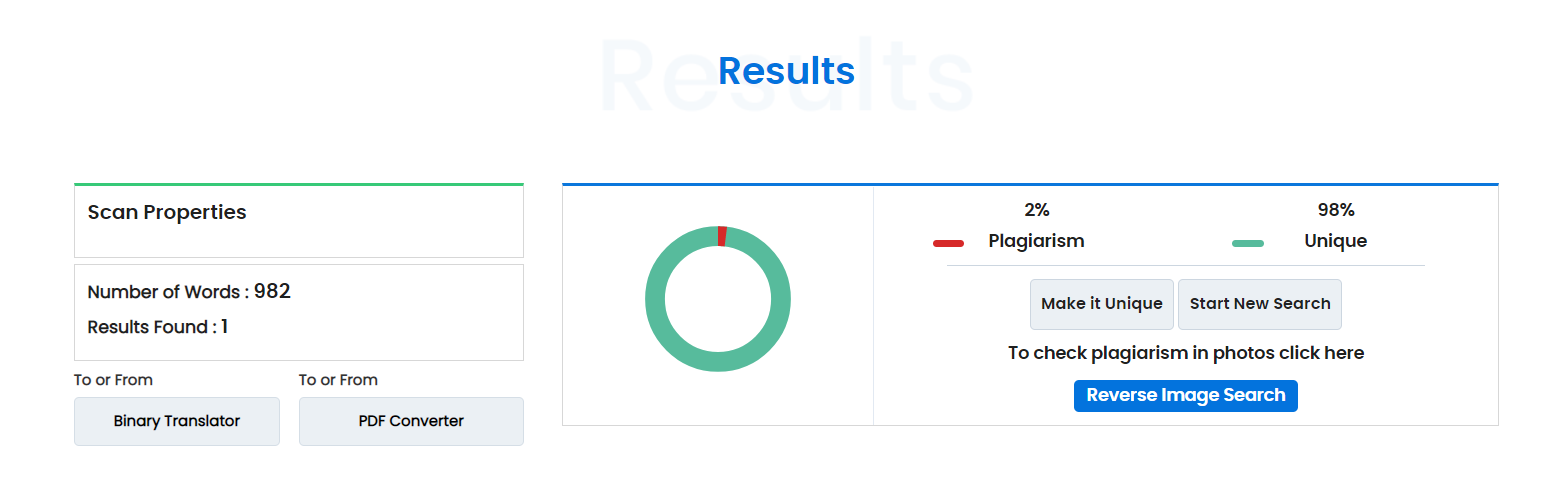
Don’t stop at positive, negative or neutral sentiment. How about noticing people at a moment of indecision? Imagine wishes, caveats, comparisons and preferences. “You should push your vendors for more than just positive, negative and neutral,” Zeszut says.

Architect for flexibility: if you built a sentiment engine 10 years ago – it won’t work today. If you don’t build for flexibility now – it will not work in 5 years. Several speakers throughout the day discussed the balance between art and science in working with big data.

“Thank you,” Zeszut concluded, “to the Googles and Twitters of the world for giving us metadata, demographic data, geodata. With each new metadata we can filter with easily, we get be more nuanced in what we offer and be more expansive about how we build the algorithms.”

|  |  |
| --- | --- |
|  |  |
|  | **Chapter - 6**  **Future Scope of Sentimental Analysis**  https://miro.medium.com/max/1400/1*IWZp8ysYNR2LlQVmTO05MA.png  **Sentiment analysis** is simply the process of categorizing the sentiments underlying a text. It is such a simple task that it can also be done manually; simply read each piece of feedback and determine whether it is positive or negative.  Among many analytical fields, one in which humans outperform all others is the ability to recognize feelings..  Despite all the challenges and potential problems that threatens Sentiment analysis, one cannot ignore the value that it adds to the industry. Because Sentiment analysis bases its results on factors that are so inherently humane, it is bound to become one the major drivers of many business decisions in future. Improved accuracy and consistency in text mining techniques can help overcome some current problems faced in Sentiment analysis. Looking ahead, what we can see is a true social democracy that will be created using Sentiment analysis, where we can harness the wisdom of the crowd rather than a select few “experts”. A democracy where every opinion counts and every sentiment affects decision making.  Sentiment Analysis has been more than just a social analytic tool. It’s been an interesting field of study. But it is a field that is still being studied, although not at great lengths due to the intricacy of this analysis. That is this field has functions that are too complicated for machines to understand. The ability to understand sarcasm, hyperbole, positive feelings, or negative feelings has been difficult, for machines that lack feelings. Algorithms have not been able to predict with more than 60% accuracy the feelings portrayed by people.  **Chapter - 7**  **Conclusion**  Working with our teammates helped us to understand what is our area of interest. we have gained an immense amount of technical knowledge from this project and we plan to continue this knowledge for my future career. we believe that my commitment as a web application developer won't solely enhance my career path however additionally we have to learn new technologies to improve my ability to create changes on our career path.  This project had its ups and downs during the development phase from those mistakes and learning we have gained so much knowledge.  This project works great for every kind of student, elderly citizen and every common man who wants to write some things down and keep them safe with us.  Also, we set ourselves some achievement list before the beginning of my internship for my improvement and I am pleased to inform that my achievement list was achieved precisely. On the other hand, we have learned to acclimate to work in a development team . As fellow programmers, we gain so much confidence dramatically due to brainstorming on this project. We are ready to work in a professional organization and eager to contribute to growth and profit with all the skills I have acquired.  **References**   * Frisklancer Consultancy Services * Learning Python by Mark Lutz * Building skills in Python by Steven F. Lott * Core Application Programming by Wesley j. Chun * [www.stackoverflow.com](http://www.stackoverflow.com) * [www.tutorialspoint.com](http://www.tutorialspoint.com) * [www.slideshare.com](http://www.slideshare.com) * [www.github.com](http://www.github.com) |

**Plagiarism Report**

****