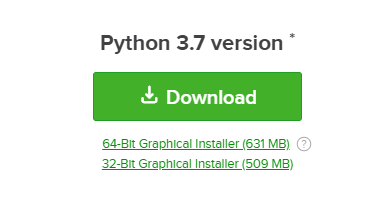
**Analisis de Sentimiento**

**Desarrollo del Laboratorio**

I – Instalación de Paquetes para reconocimiento de audio

En su computador instalar conda utilizando la versión 3.7 de python:

<https://www.anaconda.com/download/>



Luego de instalar anaconda abra una ventana de comandos y cree un ambiente virrual de desarrollo.

Esto aisla el ambiente actual de otro ambiente de desarrollo, es decir, puede crear un ambiente con paquetes diferentes para cada proyecto, de esta manera no tendrá que tener más de los paquetes requeridos.

conda create --name sentiment python=3.7.\*

Si desea ver más información de los comandos básicos de conda vaya a la siguiente dirección:

<https://conda.io/docs/user-guide/tasks/manage-environments.html>

Para activare el ambiente en la ventana de comandos ingrese: activate sentiment

Seguido instale los siguientes paquetes: pip install SpeechRecognition



II – Prueba de la instalación y código del speech recognizer en python

Descargue un audio de la siguiente página y escuche algunos segundos para entender que dice. Utilizaremos algún archivo de estos como entrada para probar el reconocimiento de audio.

<http://www.voiptroubleshooter.com/open_speech/american.html>

Abrir un archivo y guardarlo como RecognizeWAV.py o dele el nombre que ud desee.

Importe las librerías de reconocimiento de audio como sigue

**# Importa la librería de reconocimiento de audio**

**import speech\_recognition as sr**

**# Abrimos la librería o clase de speech recognizer para utilzar la recreación del modelo de audio**

**r = sr.Recognizer()**

**# Cargamos un archivo de audio. Se permiten más de una opción de reconocimiento**

**# bing, google, google cloud, houndify, pero utilizaremos google más adelante**

**myAudioFile = sr.AudioFile('OSR\_us\_000\_0017\_8k.wav')**

**# necesitamos cargar el archivo de audio como fuente para que pueda reconocer el audio**

**with myAudioFile as source:**

**audio = r.record(source)**

**# Esta línea solo imprime que el archivo es de clase audio, no tiene mayor uso, puede borrarse**

**print("Audio Type: {0}".format(type(audio)))**

**# Seguido imprimimos el resultado**

**print(r.recognize\_google(audio))**

**# También podemos descartar parte del audio, solamente acaparando el inicio utilizando la duración**

**with myAudioFile as source:**

**audio\_part = r.record(source,duration=4)**

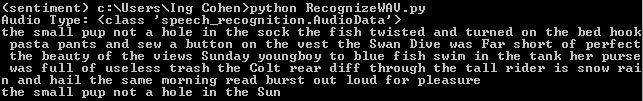
**# Reconocemos parte del audio solamente**

**print(r.recognize\_google(audio\_part))**

**print('\n\r')**

Guarde el archivo anterior. Ejecútelo en la terminal de su PC como sigue:

Python RecognizeWAV.py, debería tener una salida, dependiendo del archivo que haya utilizado como sigue:



Lo mejor sería procesar el audio en tiempo real, por consiguiente vamos a instalar los paquetes de pyAudio. En esta sección necesitamos un micrófono USB.

Para usar el micrófono instale pyAudio:

pip install PyAudio

Si existe error en la instalación utilice conda:

conda install -c anaconda pyaudio

Seguido, en la parte inferior de su archivo cree lo siguiente

**# Mensaje para avisar que puede hablar y se ejecutará el reconocimiento**

**print("Start Talking")**

**# Utilizamos el micrófono como fuente de audio**

**mic = sr.Microphone()**

**with mic as source:**

**audio = r.listen(source)**

**# Reconocemos lo que fue emitido en el microfono**

**print(r.recognize\_google(audio))**

Finalmente ejecute el archivo de python, y cuando observe que pide hablar, converse con su micrófono un par de palabras en inglés.

Sería mejor entonces que nos de una advertencia de audio si queremos realizar la grabación. Para esto utilizaremos pyttsx3

Instalar pyttsx3 en windows:

pip install pyttsx3

python –m pip install pywin32

Pruebe pyttsx3 modificando lo siguiente en el archivo RecognizeWAV.py

**import speech\_recognition as sr**

**import pyttsx3 as tts**

…

…

**print('\n\r')**

**# Arranca el reconocedor de texto**

**engine = tts.init()**

**engine.setProperty('rate', 120)**

**engine.setProperty('volume',0.9)**

**voices = engine.getProperty('voices')**

**engine.setProperty('voice', voices[1].id)**

**str = "Please start talking in english"**

**engine.say(str)**

**# Esperear a comentar**

**engine.runAndWait()**

**# detener**

**engine.stop()**

**mic = sr.Microphone()**

…

…

III – Construyendo un Analizador de Sentimiento

Esta tercera parte es analisis de sentimiento. Entramos en el desarrollo de la aplicación. Lo anterior es solo un “nice to have” o para facilitar la entrada del usuario y es más natural.

Analisis de sentimiento es una manera especial de llamar a un clasificador de texto. En este proceso determinamos en base a un fragmento de texto un valor. Por ejemplo, se puede determinar si el comentario de un usuario es positivo, negativo o neutral. Es una de las aplicaciones más comunes para procesamiento de lenguaje natural (NLP).

Adcionalmente se pueden agregar más condiciones dependiendo del problema. Se usa en la industria para medir la respuesta del usuario referente a un producto o servicio. Más allá se utiliza para campañas de mercadeo, opiniones, presencia de la compañía en redes sociales y reviews de productos de comercio electrónico. El siguiente ejemplo constará como realizar análisis de sentimiento de una película.

Podemos utilizar un clasificador cualquiera, pero ej especial uno llamado “Naive Bayes”. Lo que realiza esto es estructurar las palabras únicas de texto. El clasificador NLKT utiliza estos datos para ordenarlo de forma de diccionario para poder insertarlos. Una vez dividido estos datos en test y training set se entrenará al clasifivador para que prediga cuales comentarios son positivos y cuales son negativos. También en el primer caso imprimiremos las palabras del modelo de manera informativa indicando cuales comentarios son positivos o negativos. Estas palabras claves denotarán como reaccionará el clasfificador.

Primeramente necesitamos instalar jupyter notebook para futuros laboratorios si es necesario:

pip install jupyter notebook

Adicionalmente instalar el paquete NLTK:

pip install nltk

Y una librería de machine learning llamada scikit-learn

pip install sklearn

Lo siguiente es descargar un dataset de análisis de sentimiento de la pagina: <http://www.nltk.org/nltk_data/>

Por ejemplo movie\_reviews

Crear una carpeta llamada nlkt\_data/corpora y descomprimir los archivos que están dentro del zip “movie\_reviews”

Cree un archivo en python llamado “sentiment\_analyzer\_train.py” y pegue las siguientes líneas:

from nltk.corpus import movie\_reviews

from nltk.classify import NaiveBayesClassifier

from nltk.classify.util import accuracy as nltk\_accuracy

import pickle

Defina una función que genere un diccionario de palabras:

# Extrae las características en el diccionario de palabras

def extract\_features(words):

return dict([(word, True) for word in words])

Defina una función que cargue en este caso los comentarios de las películas:

if \_\_name\_\_=='\_\_main\_\_':

# Cargar los comentarios de la carpeta del corpus nltk\_data/corpus/movie\_reviews

fileids\_pos = movie\_reviews.fileids('pos')

fileids\_neg = movie\_reviews.fileids('neg')

Extraer las características de las películas y etiquetarlas acorde a el comentario

# Extraer las características del comentario

features\_pos = [(extract\_features(movie\_reviews.words(

fileids=[f])), 'Positive') for f in fileids\_pos]

features\_neg = [(extract\_features(movie\_reviews.words(

fileids=[f])), 'Negative') for f in fileids\_neg]

Definimos un training set de 80% y un test set de 20%

# Rompemos la información en un 80/20 training y test (80% and 20%)

threshold = 0.8

num\_pos = int(threshold \* len(features\_pos))

num\_neg = int(threshold \* len(features\_neg))

Separamos los vectores característicos del training y el test set:

# Crear el training y el test set

features\_train = features\_pos[:num\_pos] + features\_neg[:num\_neg]

features\_test = features\_pos[num\_pos:] + features\_neg[num\_neg:]

Imprimimos el número de puntos utilizados solo para referencia

# Print the number of datapoints used

print('\nNumber of training datapoints:', len(features\_train))

print('Number of test datapoints:', len(features\_test))

Entrenamos el clasificador de sentimientos

# Train a Naive Bayes classifier

classifier = NaiveBayesClassifier.train(features\_train)

print('\nAccuracy of the classifier:', nltk\_accuracy(

classifier, features\_test))

Salvamos el modelo al disco

# save the model to disk

filename = './finalized\_model.sav'

pickle.dump(classifier, open(filename, 'wb'))

print("model saved\n\r")

Solo para conocer un poco el modelo imprimimos las quince palabras más representativas

N = 15

print('\nTop ' + str(N) + ' most informative words:')

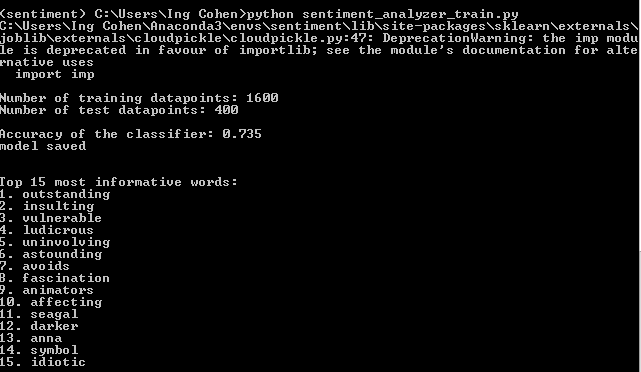
for i, item in enumerate(classifier.most\_informative\_features()):

print(str(i+1) + '. ' + item[0])

if i == N - 1:

break

Seguidamente ejecute el entrenamiento del modelo, debe de dar la siguiente salida aproximadamente:



Ahora abrir otro archivo de python y grabarlo como “sentiment\_analyzer\_test.py”

Copie las siguientes líneas que importan funciones ya recreadas

import pickle

# Extrae características

def extract\_features(words):

return dict([(word, True) for word in words])

Generamos un arreglo con información que no ha visto ni el training set ni el test set sobre comentarios de las películas.

if \_\_name\_\_=='\_\_main\_\_':

# Test input movie reviews

input\_reviews = [

'The costumes in this movie were great',

'I think the story was terrible and the characters were very weak',

'People say that the director of the movie is amazing',

'This is such an idiotic movie. I will not recommend it to anyone.'

]

Cargamos el modelo del disco:

# Cargamos el modelo

filename = './finalized\_model.sav'

classifier = pickle.load(open(filename, 'rb'))

Realizamos las predicciones de estas entradas:

print("\nMovie review predictions:")

for review in input\_reviews:

print("\nReview:", review)

# Computar probabilidades

probabilities = classifier.prob\_classify(extract\_features(review.split()))

# Tomar el valor máximo predicho

predicted\_sentiment = probabilities.max()

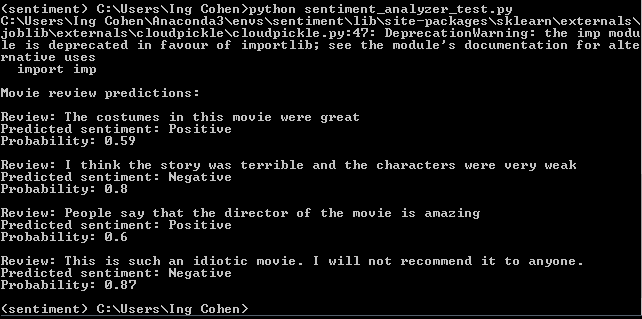
# Imprimir salidas

print("Predicted sentiment:", predicted\_sentiment)

print("Probability:", round(probabilities.prob(predicted\_sentiment), 2))

Guardamos el archivo.

Ejecute el archivo. Verá que la salida de la terminal es como sigue:



**Evaluación del Laboratorio:**

Observe la tabla de requerimientos dependiendo de su nombre, debe utilizar el clasificador y modificar el training set, además de conseguir un dataset relacionado con 100 comentarios positivos y 100 comentarios negativos mínimo. Favor bajar el dataset que provee cada uno de los sitios referente a comentarios de clientes, están disponibles en la web, de lo contrario debe crear su dataset con cada archivo de más de 500 palabras en comentarios para que haga buen trabajo de clasificación de texto.

* Todos: Completar la sección anterior: 30%
* Generar el archivo de análisis de sentimiento para que procese la entrada del comentario por audio (micrófono) y la salida del clasificador por audio (micrófono) en PC: 30%
* Transportar su modelo al raspberry pi, debe operar por micrófono: 40%

Un poco de ayuda adicional para el raspberry pi.

Instalar miniconda

wget <http://repo.continuum.io/miniconda/Miniconda3-latest-Linux-armv7l.sh>

sudo md5sum Miniconda3-latest-Linux-armv7l.sh

sudo /bin/bash Miniconda3-latest-Linux-armv7l.sh

Aceptar la instalación

Cambiar la ruta por /home/pi/miniconda3/

Editar el bashrc

Sudo nano /home/pi/.bashrc

Agregar al final

export PATH=/home/pi/miniconda3/bin:$PATH

Grabe y reincie el raspberry pi

Cree el ambiente virtual

conda create –n sentiment python=3.4.\*

conda install anaconda-client

Instalar los paquetes

Conda install SpeechRecognition pyttsx3

Instalar pyaudio espeak y cualquier paquete que necesite adicional coo libportaudio0 libportaudio2 libportcpp0 portaudio19-dev

sudo apt-get install python-dev

sudo python setup.py install

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Nombre | Test Set (%) | Modelo | Dataset |
| 1 | AGUILAR ACEVEDO GABRIEL EDUARDO | 19 | Naive Bayes Classifier | Amazon |
| 2 | BELFORD PETERS SHANTIEL AISHA | 7 | Naive Bayes Classifier | Twitter |
| 3 | CAMPOS MONTOYA KADIR - 40 | 15 | Naive Bayes Classifier | Facebook |
| 4 | CASTILLO ARAUZ ELIECER VALENTIN | 19 | Naive Bayes Classifier | Facebook |
| **5** | **CHUNG ZOU ERNESTO – 70** | **7** | **Naive Bayes Classifier** | **Pinterest** |
| 6 | FERMAN GONZALEZ ADRIAN ANTONIO | 1 | Naive Bayes Classifier | Amazon |
| **7** | **GONZALEZ BONILLA RODNEY ALEX – 70** | **19** | **Naive Bayes Classifier** | **Pinterest** |
| 8 | MEJIAS STABILITO JUAN LUIS | 5 | Naive Bayes Classifier | Instagram |
| 9 | MENDOZA LEIVA ALDAIR ANTONIO | 5 | Naive Bayes Classifier | Facebook |
| 10 | MORENO URRIOLA ROGELIO ROBERTO - 88 | 17 | Naive Bayes Classifier | Twitter |
| 11 | PEÑA ABREGO ALCIDES - 80 | 10 | Naive Bayes Classifier | Facebook |
| 12 | QIU CHEN MICHAEL 75 | 8 | Naive Bayes Classifier | Amazon |
| 13 | ROMERO RAMIREZ PEDRO ENRIQUE - 45 | 7 | Naive Bayes Classifier | Pinterest |
| 14 | RUIZ CORONADO DYLAND CHRISTIAN | 9 | Naive Bayes Classifier | Instagram |
| 15 | SALCEDO BORJA DANIELA – 60 | 15 | Naive Bayes Classifier | Linkedin |
| 16 | TELLEZ PEREZ PAOLO RICARDO - 88 | 16 | Naive Bayes Classifier | Pinterest |
| 17 | VERGARA PAREDES MIGUEL ANGEL - 75 | 7 | Naive Bayes Classifier | Amazon |
| 18 | ZAMBRANO VIDAL CESAR AUGUSTO | 4 | Naive Bayes Classifier | Instagram |
| **19** | **CHRISTIAN CRUZ – 70** | **16** | **Naive Bayes Classifier** | **Twitter** |