[ᐉ Análisis de sentimientos en Python con TextBlob | Pharos](https://pharos.sh/analisis-de-sentimientos-en-python-con-textblob/)

Los algoritmos de análisis de sentimientos se centran principalmente en definir opiniones, actitudes e incluso emoticonos en un corpus de textos. La gama de sentimientos establecidos varía significativamente de un método a otro. Mientras que un analizador estándar define hasta tres emociones polares básicas (positiva, negativa, neutra), el límite de los modelos más avanzados es más amplio.

En consecuencia, pueden mirar más allá de la polaridad y determinar seis emociones «universales» (por ejemplo, ira, disgusto, miedo, felicidad, tristeza y sorpresa):

También es posible recopilar información adicional del contexto, como el autor o un tema que, en un análisis posterior, puede prevenir un problema más complejo que una clasificación de polaridad común, a saber, subjetividad / objetividad. identificación.

Por ejemplo, esta frase de Business insider:

«En marzo, Elon Musk describió la preocupación por el brote de coronavirus como un» pánico «y» tonto «, y desde entonces tuiteó información incorrecta, como su teoría de que los niños son» esencialmente inmunes «a el virus.» expresa la subjetividad a través de una opinión personal de E. Musk, así como del autor del texto.

El enfoque que aplica el paquete TextBlob al análisis de opiniones difiere en que se basa en reglas y, por lo tanto, requiere un conjunto predefinido de palabras categorizadas. Estas palabras pueden, por ejemplo, cargarse desde la base de datos NLTK. Además, los sentimientos se definen en función de las relaciones semánticas y la frecuencia de cada palabra en una oración de entrada que permite obtener como resultado una salida más precisa.

Una vez que se logra el primer paso y un modelo de Python se alimenta con los datos de entrada necesarios, un usuario puede obtener los puntajes de sentimiento en forma de polaridad y subjetividad que se discutieron en la sección anterior. Podemos ver cómo funciona este proceso en este artículo al [Foro Kapadia](https://www.cs.rit.edu/usr/local/pub/GraduateProjects/2165/fjk9481/Report.pdf):

Salida de TextBlob para un **polaridad** la tarea es un flotador dentro del rango [-1.0, 1.0] dónde -1.0 es una polaridad negativa y 1.0 es positivo. Esta puntuación también puede ser igual a 0, que representa una evaluación neutral de una declaración, ya que no contiene ninguna palabra del conjunto de entrenamiento.

Considerando que, un **subjetividad / objetividad** la tarea de identificación informa un flotador dentro del rango [0.0, 1.0]

Otra opción

# Importing TextBlob

from textblob import TextBlob

# Preparing an input sentence

sentence="The platform provides universal access to the world's best education, partnering with top universities and organizations to offer courses online."

# Creating a textblob object and assigning the sentiment property

analysis = TextBlob(sentence).sentiment

print(analysis)

output

Sentiment(polarity=0.5, subjectivity=0.26666666666666666)

O por separado

analysisPol = TextBlob(sentence).polarity

analysisSub = TextBlob(sentence).subjectivity

print(analysisPol)

print(analysisSub)

0.5

0.26666666666666666

I am sure that is the reason why education is so important.

Sentiment(polarity=0.45, subjectivity=0.944)

Later, do not stop to sharing knowledge with others.

Sentiment(polarity=0.0, subjectivity=0.0)

Una de las mejores cosas de TextBlob es que permite al usuario elegir un algoritmo para la implementación de las tareas de PNL de alto nivel:

* PatternAnalyzer – un clasificador predeterminado que se basa en la biblioteca de patrones
* NaiveBayesAnalyzer – un modelo NLTK entrenado en un corpus de reseñas de películas

Para cambiar la configuración predeterminada, simplemente especificaremos un NaiveBayes analizador en el código. Ejecutemos un análisis de sentimiento en tweets directamente desde [Gorjeo](https://twitter.com/):

from textblob import TextBlob

# For parsing tweets

import tweepy

# Importing the NaiveBayesAnalyzer classifier from NLTK

from textblob.sentiments import NaiveBayesAnalyzer

Después de eso, necesitamos establecer una conexión con la API de Twitter a través de claves API (que puede obtener a través de un [cuenta de desarrollador](https://developer.twitter.com/en)):

# Uploading api keys and tokens

api\_key = 'XXXXXXXXXXXXXXX'

api\_secret="XXXXXXXXXXXXXXX"

access\_token = 'XXXXXXXXXXXXXXX'

access\_secret="XXXXXXXXXXXXXXX"

# Establishing the connection

twitter = tweepy.OAuthHandler(api\_key, api\_secret)

api = tweepy.API(twitter)

Ahora, podemos realizar el análisis de tweets sobre cualquier tema. Una palabra buscada (por ejemplo, bloqueo) puede ser una palabra o más. Además, esta tarea puede llevar mucho tiempo debido a la enorme cantidad de tweets. Se recomienda limitar la salida:

# This command will call back 5 tweets within a “lockdown” topic

corpus\_tweets = api.search("lockdown", count=5)

for tweet in corpus\_tweets:

print(tweet.text)

El resultado de este último fragmento de código traerá cinco tweets que mencionan la palabra buscada de la siguiente forma:

[email protected]: How Asia's densest slum contained the virus and the economic catastrophe that stares at the hardworking slum population...

El último paso en este ejemplo es cambiar el modelo predeterminado al analizador NLTK que devuelve sus resultados como un namedtuple de la forma: Sentiment(classification, p\_pos, p\_neg):

# Applying the NaiveBayesAnalyzer

blob\_object = TextBlob(tweet.text, analyzer=NaiveBayesAnalyzer())

# Running sentiment analysis

analysis = blob\_object.sentiment

print(analysis)

Finalmente, nuestro modelo de Python nos dará la siguiente evaluación de sentimiento:

Sentiment(classification='pos', p\_pos=0.5057908299783777, p\_neg=0.49420917002162196)

Aquí, lo clasifica como un sentimiento positivo, con el p\_pos y p\_neg los valores son ~0.5 cada.