**Análisis de sentimientos con Python**



**Adrián Yared Armas de la Nuez**

**Contenido**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

[**1. Actividad 2**](#_hs8tgj1rv02a)

[**1.2. Enunciado 2**](#_mbqaeu8copk6)

[**1.3. Resolución 2**](#_nqkhgsgt6zf5)

[**2. Actividad 2**](#_6t6o438fh1)

[**2.2. Enunciado general 2**](#_kds84iuf9o0p)

[**2.2.1.1 Enunciado actividad 1 2**](#_a3d4o389z5or)

[**2.2.1.2 Resolución 3**](#_mcbq8c2b5n38)

[**2.2.2.1 Enunciado actividad 2 3**](#_tmtwqr6v34yj)

[**2.2.2.2.2 Actividad del artículo 1 3**](#_ve50muuzv5lm)

[**2.2.2.2.3 Actividad del artículo 2 6**](#_3ranat40e6t5)

[**2.2.2.2.4 Actividad del artículo 3 8**](#_hwo8qvv5qsji)

[**2.2.2.2.5 Actividad del artículo 4 9**](#_vzuce87ptis)

[**2.2.2.2.6 Actividad del artículo 5 13**](#_lwyzxgnj64n2)

[**3. Enlace del colab 14**](#_fjt1qj6ljwf1)

## **1. Actividad**

### **1.2. Enunciado**

Resuma el artículo a continuación, destacando los aspectos clave y las conclusiones principales: <https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17>

### **1.3. Resolución**

El artículo explica la importancia del análisis de sentimientos, que es una técnica de minería de texto que permite a las empresas comprender la percepción social sobre sus marcas. El autor destaca que el análisis básico de sentimientos, como la clasificación de mensajes en positivos, negativos o neutrales, es superficial y pierde perspectivas más profundas que pueden ser extraídas con tecnologías avanzadas como el aprendizaje profundo y la inteligencia artificial.

Los aspectos claves principales a destacar según mi criterio son las siguientes; la definición de análisis de sentimientos; el análisi de intenciones, que va más allá del análisis de sentimientos, permitiendo identificar la intención del mensaje; la búsqueda css (semántica contextual), esto clasifica los mensajes según los contextos sin depender de palabras clave exactas; y finalmente el caso de Estudio y ejemplo de Uber, que realizó un análisis profundo de las redes sociales sobre los comentarios sobre Uber y sus determinadas categorías (Precios, seguridad, pagos, etc). Este análisis permitió a la empresa darse cuenta de problemas con cancelaciones y con el servicio que permitieron ponerles alerta y mejorar su servicio.

## **2. Actividad**

### **2.2. Enunciado general**

Verifique que la información mostrada en este artículo coincide con el artículo de la actividad R.1 de análisis de sentimientos:

https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/

### **2.2.1.1 Enunciado actividad 1**

Lleve a cabo un proceso de resumen, destacando las ideas principales y realice una comparación con el artículo del ejercicio anterior.

( [Artículo 1](https://www.analyticsvidhya.com/blog/2022/07/sentiment-analysis-using-python/) )( [Artículo 2](https://towardsdatascience.com/sentiment-analysis-concept-analysis-and-applications-6c94d6f58c17) )

### **2.2.1.2 Resolución**

Este nuevo artículo explica cómo realizar análisis de sentimientos en Python, una técnica clave en el procesamiento de lenguaje natural para clasificar textos según sus sentimientos y se detallan diversas herramientas como; TextBlob, que es una librería que devuelve valores de polaridad y subjetividad; VADER, un analizador basado en el uso de las redes sociales; Modelos de aprendizaje automático; modelos basados en LSTM (Modelos de redes neuronales recurrentes para datos secuenciales); y modelos basados en transformaciones, como el uso de modelos pre entrenados

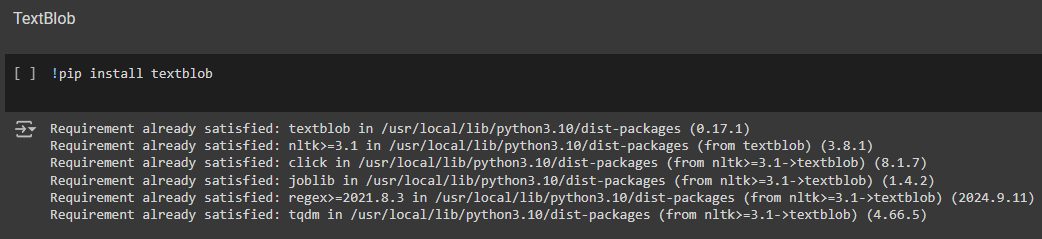
Ambos artículos abordan el análisis de sentimientos, pero con enfoques y profundidades diferentes:

El nuevo artículo presenta una explicación mucho más detallada sobre el uso de herramientas para el análisis de sentimientos, sin embargo el primer artículo muestra los principios básicos con menos profundidad técnica. Además, el segundo artículo muestra ejemplos detallados y un enfoque práctico basado principalmente en python, mientras que el artículo anterior menciona las aplicaciones generales de manera más abstracta resaltando su simplicidad y variedad de métodos.

### **2.2.2.1 Enunciado actividad 2**

Adicionalmente, verifique el funcionamiento de los códigos de Python presentados en el artículo. Es fundamental que aumente el tamaño de la muestra en cada uno de los ejemplos para obtener resultados más consistentes.

### **2.2.2.2.2 Actividad del artículo 1**

Instalación de textblob:  


Código con más ejemplos:

from textblob import TextBlob # import TextBlob

# main.py

from textblob import TextBlob

# Definiendo una lista de ejemplos de texto

texts = [

"The movie was so awesome.", # positivo

"The food here tastes terrible.", # negativo

"I love this place!", # positivo

"This is the worst service ever.", # negativo

"What a fantastic experience!", # positivo

"I'm so disappointed with my order.",# negativo

"The view from the top is breathtaking.", # positivo

"I hate waiting in long lines.", # negativo

"This book is a masterpiece.", # positivo

"I can't stand the noise in the city.", # negativo

"The weather is beautiful today.", # positivo

"I feel so sad and alone.", # negativo

"This product exceeded my expectations!", # positivo

"The customer service was unhelpful.", # negativo

"The concert was absolutely thrilling!", # positivo

"I'm frustrated with my computer.", # negativo

"I would recommend this restaurant to everyone!", # positivo

"The film was a complete disaster.", # negativo

"Her performance was mesmerizing.", # positivo

"I am not satisfied with the results." # negativo

]

# Inicializando listas para polaridad y subjetividad

polarities = []

subjectivities = []

# Procesando cada texto

for i, text in enumerate(texts):

# Determinando la polaridad

polarity = TextBlob(text).sentiment.polarity

polarities.append(polarity)

# Determinando la subjetividad

subjectivity = TextBlob(text).sentiment.subjectivity

subjectivities.append(subjectivity)

print(f"Text {i+1}: '{text}'")

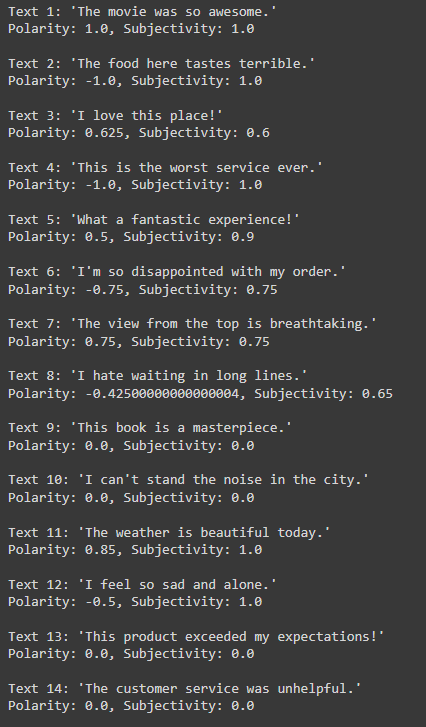
print(f"Polarity: {polarity}, Subjectivity: {subjectivity}\n")

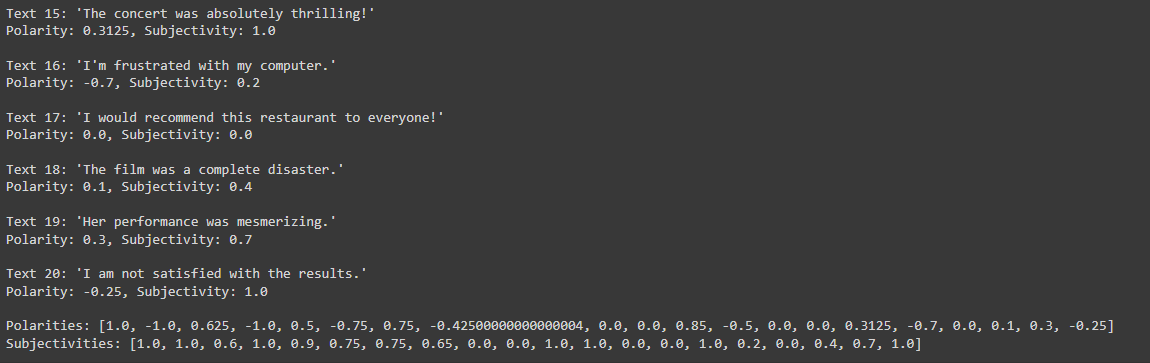
# Mostrando resultados finales

print("Polarities:", polarities)

print("Subjectivities:", subjectivities)

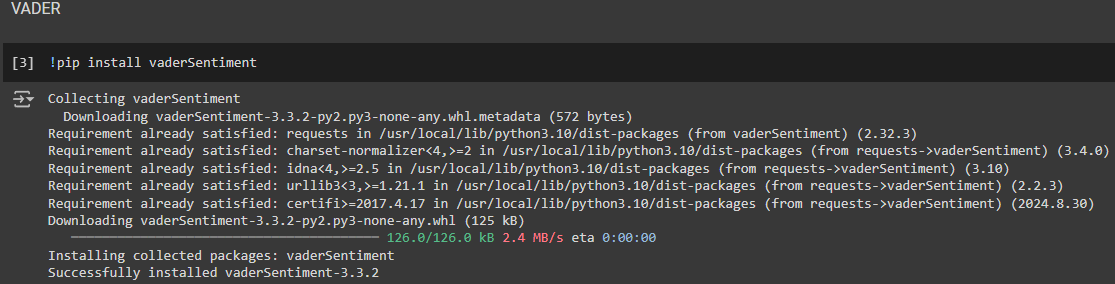
Resultado de ejecución con más ejemplos:





### **2.2.2.2.3 Actividad del artículo 2**

Instalación de Vader:



Código con más ejemplos:

from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer

# Inicializando el analizador de sentimientos

sentiment = SentimentIntensityAnalyzer()

# Definiendo una lista de ejemplos de texto

texts = [

"The book was a perfect balance between writing style and plot.", # positivo

"The pizza tastes terrible.", # negativo

"I absolutely loved the new movie!", # positivo

"The service was awful and very slow.", # negativo

"This is one of the best vacations I've ever had.", # positivo

"I'm so unhappy with my experience here.", # negativo

"What a delightful surprise to find this cafe!", # positivo

"The noise outside is really annoying.", # negativo

"Her artwork is incredibly beautiful.", # positivo

"I'm frustrated with the long wait times.", # negativo

"The atmosphere in this restaurant is amazing.", # positivo

"This product did not meet my expectations.", # negativo

"The concert was unforgettable and magical.", # positivo

"I'm not satisfied with my purchase at all.", # negativo

"He gave an inspiring speech that motivated everyone.", # positivo

"I feel very disappointed by the lack of options.", # negativo

"The new restaurant in town has delicious food!", # positivo

"I cannot recommend this hotel due to its poor cleanliness.", # negativo

"The weather was perfect for a day at the beach.", # positivo

"I am very upset about the recent changes.", # negativo

"This book is a real page-turner and keeps you engaged!", # positivo

"The quality of the service was subpar and not worth the price." # negativo

]

# Procesando cada texto y mostrando el sentimiento

for i, text in enumerate(texts):

# Obteniendo los puntajes de polaridad

sentiment\_scores = sentiment.polarity\_scores(text)

print(f"Sentiment of text {i + 1}: '{text}' -> {sentiment\_scores}\n")

Resultado de ejecución con más ejemplos:



### **2.2.2.2.4 Actividad del artículo 3**

Código para el análisis de sentimientos utilizando el enfoque de vectorización de bolsa de palabras

código:  
#Loading the Dataset

import pandas as pd

data = pd.read\_csv('data.csv')

#Pre-Prcoessing and Bag of Word Vectorization using Count Vectorizer

from sklearn.feature\_extraction.text import CountVectorizer

from nltk.tokenize import RegexpTokenizer

token = RegexpTokenizer(r'[a-zA-Z0-9]+')

cv = CountVectorizer(stop\_words='english',ngram\_range = (1,1),tokenizer = token.tokenize)

text\_counts = cv.fit\_transform(data['Sentence'])

#Splitting the data into trainig and testing

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(text\_counts, data['Sentiment'], test\_size=0.25, random\_state=5)

#Training the model

from sklearn.naive\_bayes import MultinomialNB

MNB = MultinomialNB()

MNB.fit(X\_train, Y\_train)

#Caluclating the accuracy score of the model

from sklearn import metrics

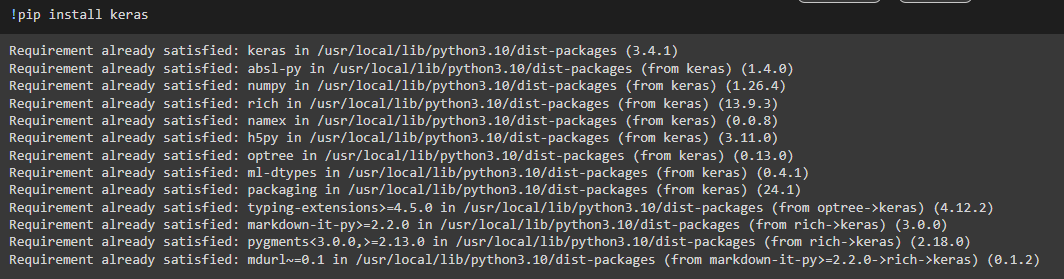
predicted = MNB.predict(X\_test)

accuracy\_score = metrics.accuracy\_score(predicted, Y\_test)

print("Accuracuy Score: ",accuracy\_score)

Resultado:  


### **2.2.2.2.5 Actividad del artículo 4**

Instalación Keras:  


código:  
# Importing necessary libraries

import nltk

import pandas as pd

from textblob import Word

from nltk.corpus import stopwords

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import classification\_report, confusion\_matrix, accuracy\_score

from keras.models import Sequential

from tensorflow.keras.layers import LeakyReLU

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D

from sklearn.model\_selection import train\_test\_split

# Download the necessary stopwords dataset

nltk.download('stopwords') # Downloads English stopwords

nltk.download('wordnet') # Downloads WordNet lexicon for lemmatization

nltk.download('omw-1.4') # Downloads additional WordNet lexicon

# Loading the dataset

data = pd.read\_csv('data.csv') # Loads the dataset from a CSV file

# Pre-Processing the text

def cleaning(df, stop\_words):

# Lowercasing the sentence

df['Sentence'] = df['Sentence'].apply(lambda x: ' '.join(x.lower() for x in x.split())) # Converts to lowercase

# Replacing digits/numbers

df['Sentence'] = df['Sentence'].str.replace('\d+', '', regex=True) # Removes digits from the text

# Removing stop words

df['Sentence'] = df['Sentence'].apply(lambda x: ' '.join(x for x in x.split() if x not in stop\_words)) # Removes stop words

# Lemmatization

df['Sentence'] = df['Sentence'].apply(lambda x: ' '.join([Word(word).lemmatize() for word in x.split()])) # Applies lemmatization to words

return df # Returns the processed DataFrame

# Get the list of stop words in English

stop\_words = stopwords.words('english')

# Clean the dataset

data\_cleaned = cleaning(data, stop\_words)

# Generating Embeddings using tokenizer

tokenizer = Tokenizer(num\_words=500, split=' ')

tokenizer.fit\_on\_texts(data\_cleaned['Sentence'].values)

# Defining X and y

X = tokenizer.texts\_to\_sequences(data\_cleaned['Sentence'].values)

X = pad\_sequences(X)

# Convert labels to one-hot encoding

y = pd.get\_dummies(data\_cleaned['Sentiment']).values

# Model Building

model = Sequential()

model.add(Embedding(500, 120, input\_length=X.shape[1]))

model.add(SpatialDropout1D(0.4))

model.add(LSTM(704, dropout=0.2, recurrent\_dropout=0.2))

model.add(Dense(352))

model.add(LeakyReLU(negative\_slope=0.1))

model.add(Dense(3, activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy']) # Compile

print(model.summary())

# Splitting the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Model Training

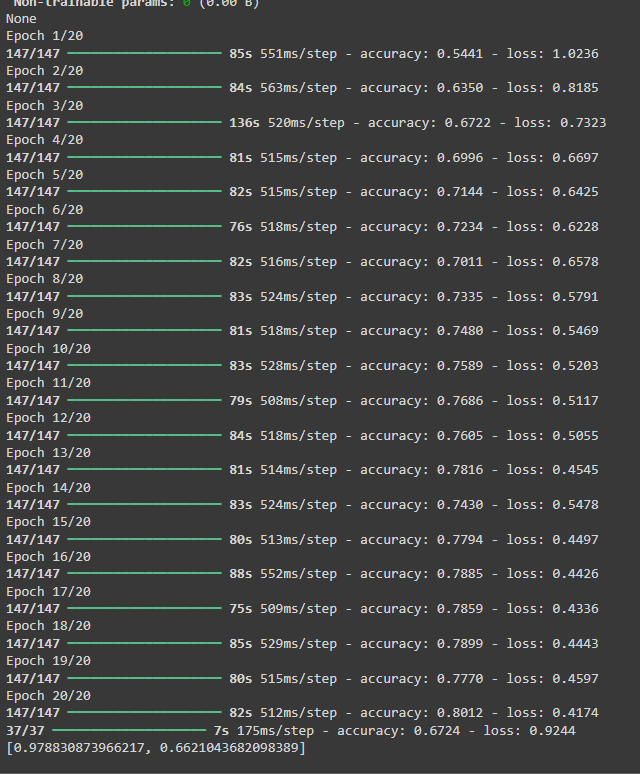
model.fit(X\_train, y\_train, epochs=20, batch\_size=32, verbose=1)

# Model Testing

model.evaluate(X\_test, y\_test)

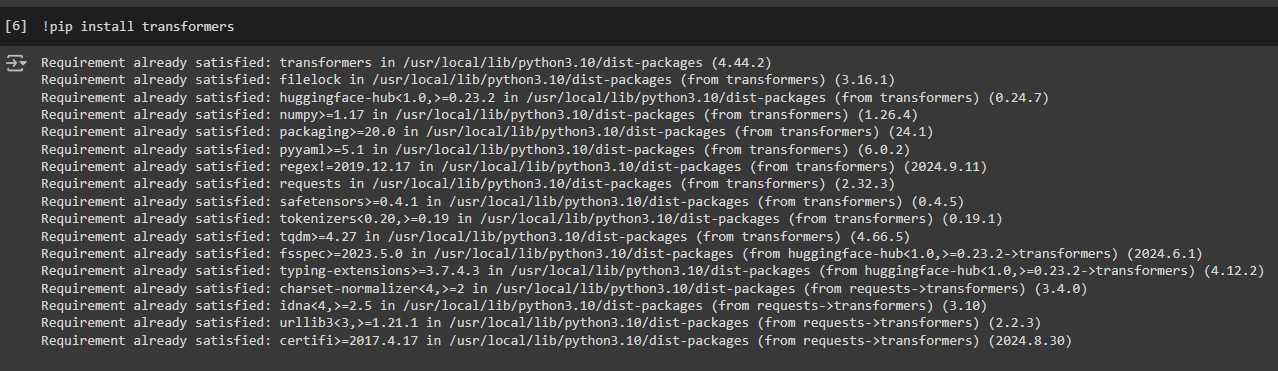
Resultado:

Modificar librerías y nombres de archivos o columnas en un modelo LSTM puede causar incompatibilidades con nuevas versiones de Keras y TensorFlow, dificultar la reproducibilidad y el mantenimiento, e introducir errores en la ejecución. Es ideal documentar estos cambios, fijar versiones y usar archivos de configuración para asegurar estabilidad y flexibilidad en el modelo.



### 

### **2.2.2.2.6 Actividad del artículo 5**

Instalación de transformers:  


Código con más ejemplos:  
import transformers

from transformers import pipeline

# Inicializando el pipeline de análisis de sentimientos

sentiment\_pipeline = pipeline("sentiment-analysis")

# Definiendo una lista de ejemplos de texto

data = [

"It was the best of times.", # positivo

"It was the worst of times.", # negativo

"I absolutely love this place!", # positivo

"The service here is terrible.", # negativo

"What an amazing experience!", # positivo

"I'm so disappointed with the movie.", # negativo

"The view from the top is breathtaking.", # positivo

"I can't stand the noise from the street.", # negativo

"This book is a fantastic read!", # positivo

"The food was bland and uninspiring.", # negativo

"Her performance was outstanding!", # positivo

"I feel very let down by this product.", # negativo

"The weather is perfect for a picnic.", # positivo

"I'm really frustrated with the delay.", # negativo

"This restaurant serves the best pasta I've ever had!", # positivo

"I'm unhappy with how my complaint was handled.", # negativo

"The concert was exhilarating!", # positivo

"I don't think I will return to this place.", # negativo

"The movie had a beautiful soundtrack.", # positivo

"I felt bored and restless during the presentation.", # negativo

"This app has changed my life for the better!" # positivo

]

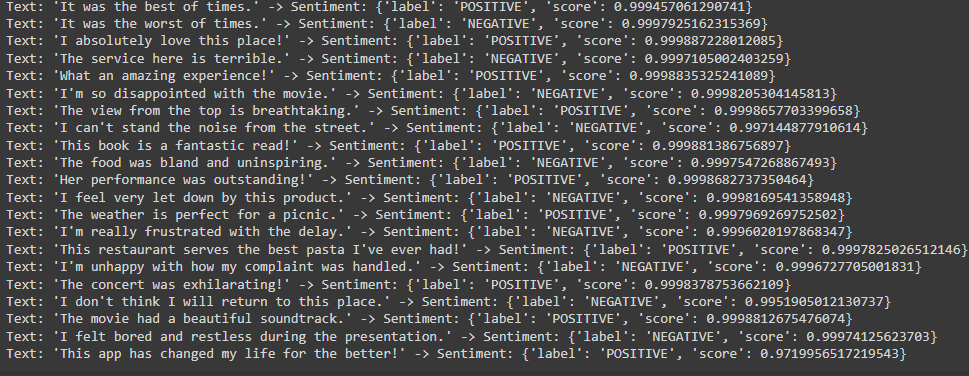
# Realizando el análisis de sentimientos

results = sentiment\_pipeline(data)

# Imprimiendo los resultados

for i, text in enumerate(data):

print(f"Text: '{text}' -> Sentiment: {results[i]}")

Resultado:  


## **3. Enlace del colab**

[](https://colab.research.google.com/drive/1Z-KUkJCEzxE1XCVJXWS1YPt9-yHOAOfz?usp=sharing)