RESUMEN

Análisis de sentimientos con Python

Resumen

En el artículo citado se habla sobre las grandes cantidades de datos que hay en las redes sociales y cómo usarlos para fines comerciales mediante el análisis de sentimientos o también llamado minería de opiniones.

Ideas principales

El artículo se centra en explicar que es el análisis de sentimientos, que tipos hay y pone ejemplos para explicarlo:

- **¿Cómo funciona el análisis de sentimiento?**
  El análisis de sentimiento funciona empleando técnicas de procesamiento del lenguaje natural (PLN). El proceso consta de varios pasos:
    - Preprocesamiento del texto
    - Tokenización
    - Extracción de características
    - Clasificación del sentimiento
    - Post-procesamiento
    - Evaluación
- **Tipos de análisis de sentimiento**
    - Análisis del sentimiento a nivel de documento
    - Análisis del sentimiento a nivel de frase
    - Análisis de sentimiento basado en aspectos
    - Análisis de sentimiento a nivel de entidad
    - Análisis comparativo del sentimiento
- **Casos prácticos de análisis de sentimiento**
    - Monitorización de redes sociales para la gestión de marcas
    - Análisis de productos/servicios
    - Predicción del precio de las acciones
- **Formas de realizar análisis de sentimiento en Python**
    - Uso del Bloque de Texto
    - Usando a Vader

- Uso de modelos basados en la vectorización de bolsas de palabras
- Uso de modelos basados en LSTM
- Utilización de modelos basados en transformadores

Conclusión

Tenemos grandes cantidades de datos en las redes sociales que podemos explotar haciendo uso del análisis de sentimientos con la herramienta que más nos convenga para nuestro caso de uso.

Comparativa con el artículo anterior

En comparación con el artículo anterior este es más didáctico, te explica muy bien que es el análisis de sentimientos, como se usa y ejemplos de cómo usarlos en Python, para tener una idea de cuan útil es y cómo se podría aplicar a los casos de uso que se le pueda ocurrir al lector.

El anterior artículo se centraba más en explicar que es el análisis de sentimientos y lo útil que sería en ciertos casos de usos, pero no te explicaba bien su funcionamiento para que el lector ya pudieran experimentar cómo se usan estas herramientas.

En conclusión, los dos artículos están bastante bien, el anterior tiene una visión más comercial, para que los lectores entiendan que es el análisis de sentimientos y puedan empezar a investigar más en cómo usar esta herramienta y el actual es más didáctico, a parte de explicarte que es el análisis de sentimientos, te pone ejemplos que podrían ser reales y te explica las diferentes formas de realizar el análisis en Python.

Ejecución de código

Usando Text Blob

Iniciando modelo y definir el dataSet

```python
from textblob import TextBlob

text_1 = "The movie was so awesome."
text_2 = "The food here tastes terrible."
text_3 = "The sunrise painted the sky with breathtaking hues."
text_4 = "I regret purchasing this product; it doesn't meet my expectations."
text_5 = "The traffic jam turned my morning commute into a nightmare."
text_6 = "The plain white walls of the room create a minimalist aesthetic."
text_7 = "Her kindness and generosity know no bounds."
text_8 = "I aced my exam and couldn't be happier."
text_9 = "The customer service representative was rude and unhelpful."
text_10 = "The textbook provides a comprehensive overview of the subject matter."
text_11 = "The email contained important information about the upcoming event."
text_12 = "Spending quality time with loved ones always brings immense joy."
text_13 = "The unexpected compliment brightened my entire day."
```

Determinar polaridad y subjetividad

```python
#Determining the Polarity
p_1 = TextBlob(text_1).sentiment.polarity
p_2 = TextBlob(text_2).sentiment.polarity
p_3 = TextBlob(text_3).sentiment.polarity
p_4 = TextBlob(text_4).sentiment.polarity
p_5 = TextBlob(text_5).sentiment.polarity
p_6 = TextBlob(text_6).sentiment.polarity
p_7 = TextBlob(text_7).sentiment.polarity
p_8 = TextBlob(text_8).sentiment.polarity
p_9 = TextBlob(text_9).sentiment.polarity
p_10 = TextBlob(text_10).sentiment.polarity
p_11 = TextBlob(text_11).sentiment.polarity
p_12 = TextBlob(text_12).sentiment.polarity
p_13 = TextBlob(text_13).sentiment.polarity

#Determining the Subjectivity
s_1 = TextBlob(text_1).sentiment.subjectivity
s_2 = TextBlob(text_2).sentiment.subjectivity
s_3 = TextBlob(text_3).sentiment.subjectivity
s_4 = TextBlob(text_4).sentiment.subjectivity
s_5 = TextBlob(text_5).sentiment.subjectivity
s_6 = TextBlob(text_6).sentiment.subjectivity
s_7 = TextBlob(text_7).sentiment.subjectivity
s_8 = TextBlob(text_8).sentiment.subjectivity
s_9 = TextBlob(text_9).sentiment.subjectivity
s_10 = TextBlob(text_10).sentiment.subjectivity
s_11 = TextBlob(text_11).sentiment.subjectivity
s_12 = TextBlob(text_12).sentiment.subjectivity
s_13 = TextBlob(text_13).sentiment.subjectivity
```

Imprimir resultados

```
print("Polarity of Text 1 is", p_1)
print("Polarity of Text 2 is", p_2)
print("Polarity of Text 3 is ", p_3)
print("Polarity of Text 4 is ", p_4)
print("Polarity of Text 5 is ", p_5)
print("Polarity of Text 6 is ", p_6)
print("Polarity of Text 7 is ", p_7)
print("Polarity of Text 8 is ", p_8)
print("Polarity of Text 9 is ", p_9)
print("Polarity of Text 10 is ", p_10)
print("Polarity of Text 11 is ", p_11)
print("Polarity of Text 12 is ", p_12)
print("Polarity of Text 13 is ", p_13)
print("Subjectivity of Text 1 is", s_1)
print("Subjectivity of Text 2 is", s_2)
print("Subjectivity of Text 3 is ", s_3)
print("Subjectivity of Text 4 is ", s_4)
print("Subjectivity of Text 5 is ", s_5)
print("Subjectivity of Text 6 is ", s_6)
print("Subjectivity of Text 7 is ", s_7)
print("Subjectivity of Text 8 is ", s_8)
print("Subjectivity of Text 9 is ", s_9)
print("Subjectivity of Text 10 is ", s_10)
print("Subjectivity of Text 11 is ", s_11)
print("Subjectivity of Text 12 is ", s_12)
print("Subjectivity of Text 13 is ", s_13)
```

Resultados

```
Polarity of Text 1 is 1.0
Polarity of Text 2 is -1.0
Polarity of Text 3 is  1.0
Polarity of Text 4 is  0.0
Polarity of Text 5 is  0.0
Polarity of Text 6 is  -0.10714285714285714
Polarity of Text 7 is  0.0
Polarity of Text 8 is  0.0
Polarity of Text 9 is  -0.3
Polarity of Text 10 is  -0.16666666666666666
Polarity of Text 11 is  0.4
Polarity of Text 12 is  0.5
Polarity of Text 13 is  0.05
Subjectivity of Text 1 is 1.0
Subjectivity of Text 2 is 1.0
Subjectivity of Text 3 is  1.0
Subjectivity of Text 4 is  0.0
Subjectivity of Text 5 is  0.0
Subjectivity of Text 6 is  0.17857142857142858
Subjectivity of Text 7 is  0.0
Subjectivity of Text 8 is  0.0
Subjectivity of Text 9 is  0.6
Subjectivity of Text 10 is  0.3333333333333333
Subjectivity of Text 11 is  1.0
Subjectivity of Text 12 is  0.6666666666666666
Subjectivity of Text 13 is  0.8125
```

Usando Vader

Iniciando modelo y definir el dataSet

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
sentiment = SentimentIntensityAnalyzer()
text_1 = "The book was a perfect balance between wrtiting style and plot."
text_2 =  "The pizza tastes terrible."
text_3 = "The sunrise painted the sky with breathtaking hues."
text_4 = "I regret purchasing this product; it doesn't meet my expectations."
text_5 = "The traffic jam turned my morning commute into a nightmare."
text_6 = "The plain white walls of the room create a minimalist aesthetic."
text_7 = "Her kindness and generosity know no bounds."
text_8 = "I aced my exam and couldn't be happier."
text_9 = "The customer service representative was rude and unhelpful."
text_10 = "The textbook provides a comprehensive overview of the subject matter."
text_11 = "The email contained important information about the upcoming event."
text_12 = "Spending quality time with loved ones always brings immense joy."
text_13 = "The unexpected compliment brightened my entire day."
```

Determinar subjetividad

```
sent_1 = sentiment.polarity_scores(text_1)
sent_2 = sentiment.polarity_scores(text_2)
sent_3 = sentiment.polarity_scores(text_3)
sent_4 = sentiment.polarity_scores(text_4)
sent_5 = sentiment.polarity_scores(text_5)
sent_6 = sentiment.polarity_scores(text_6)
sent_7 = sentiment.polarity_scores(text_7)
sent_8 = sentiment.polarity_scores(text_8)
sent_9 = sentiment.polarity_scores(text_9)
sent_10 = sentiment.polarity_scores(text_10)
sent_11 = sentiment.polarity_scores(text_11)
sent_12 = sentiment.polarity_scores(text_12)
sent_13 = sentiment.polarity_scores(text_13)
```

Imprimir resultados

```
print("Sentiment of text 1:", sent_1)
print("Sentiment of text 2:", sent_2)
print("Sentiment of text 3:", sent_3)
print("Sentiment of text 4:", sent_4)
print("Sentiment of text 5:", sent_5)
print("Sentiment of text 6:", sent_6)
print("Sentiment of text 7:", sent_7)
print("Sentiment of text 8:", sent_8)
print("Sentiment of text 9:", sent_9)
print("Sentiment of text 10:", sent_10)
print("Sentiment of text 11:", sent_11)
print("Sentiment of text 12:", sent_12)
print("Sentiment of text 13:", sent_13)
```

Resultados

```
Sentiment of text 1: {'neg': 0.0, 'neu': 0.73, 'pos': 0.27, 'compound': 0.5719}
Sentiment of text 2: {'neg': 0.508, 'neu': 0.492, 'pos': 0.0, 'compound': -0.4767}
Sentiment of text 3: {'neg': 0.0, 'neu': 0.7, 'pos': 0.3, 'compound': 0.4588}
Sentiment of text 4: {'neg': 0.237, 'neu': 0.763, 'pos': 0.0, 'compound': -0.4215}
Sentiment of text 5: {'neg': 0.0, 'neu': 1.0, 'pos': 0.0, 'compound': 0.0}
Sentiment of text 6: {'neg': 0.0, 'neu': 0.826, 'pos': 0.174, 'compound': 0.2732}
Sentiment of text 7: {'neg': 0.176, 'neu': 0.32, 'pos': 0.504, 'compound': 0.6249}
Sentiment of text 8: {'neg': 0.284, 'neu': 0.716, 'pos': 0.0, 'compound': -0.4168}
Sentiment of text 9: {'neg': 0.3, 'neu': 0.7, 'pos': 0.0, 'compound': -0.4588}
Sentiment of text 10: {'neg': 0.0, 'neu': 0.721, 'pos': 0.279, 'compound': 0.2732}
Sentiment of text 11: {'neg': 0.0, 'neu': 0.816, 'pos': 0.184, 'compound': 0.2023}
Sentiment of text 12: {'neg': 0.0, 'neu': 0.51, 'pos': 0.49, 'compound': 0.8271}
Sentiment of text 13: {'neg': 0.0, 'neu': 0.446, 'pos': 0.554, 'compound': 0.7351}
```

Usando el modelo Bag of Words Vectorization-Based

Cargar el dataSet

```
#Loading the Dataset
import pandas as pd
data = pd.read_csv('data.csv')
✓ 0.7s
```

Preprocesamiento

```
#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'])
✓ 0.0s
```

Parametrización e inserción de datos de entrenamiento

```
#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=0.02, random_state=100)
✓ 0.0s
```

Entrenamiento

```
#Training the model
from sklearn.naive_bayes import MultinomialNB
MNB = MultinomialNB()
MNB.fit(X_train, Y_train)
✓ 0.0s

▾ MultinomialNB
MultinomialNB()
```

Cálculo y resultado

```
#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)
```

0.0s

```
Accuracuy Score:  0.7350427350427351
```

Usando el modelo LSTM-Based

Importamos las librerías necesarias

```
#Importing necessary libraries
import nltk
import pandas as pd
from textblob import Word
nltk.download('stopwords')
nltk.download('wordnet')
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
from keras.models import Sequential
from keras.utils import to_categorical
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from keras.layers import Dense, Embedding, LSTM, SpatialDropout1D
from sklearn.model_selection import train_test_split
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Cargamos el dataSet

```
#Loading the dataset
from google.colab import drive

drive.mount('/content/drive')
data = pd.read_csv('/content/drive/MyDrive/Master FP IA y BD/SPS/Análisis de sentimientos con Python/code/data.csv')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Pre-procesamos los datos

```
#Pre-Processing the text
def cleaning(df, stop_words):
    df['Sentence'] = df['Sentence'].apply(lambda x: ' '.join(x.lower() for x in x.split()))
    # Replacing the digits/numbers
    df['Sentence'] = df['Sentence'].str.replace('d', '')
    # Removing stop words
    df['Sentence'] = df['Sentence'].apply(lambda x: ' '.join(x for x in x.split() if x not in stop_words))
    # Lemmatization
    df['Sentence'] = df['Sentence'].apply(lambda x: ' '.join([Word(x).lemmatize() for x in x.split()]))
    return df
stop_words = stopwords.words('english')
data_cleaned = cleaning(data, stop_words)
```

Generamos los Embeddings

```
[65] #Generating Embeddings using tokenizer
tokenizer = Tokenizer(num_words=500, split=' ')
tokenizer.fit_on_texts(data_cleaned['Sentence'].values)
X = tokenizer.texts_to_sequences(data_cleaned['Sentence'].values)
X = pad_sequences(X)
```

Construimos el modelo

```
#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, activation='LeakyReLU'))
model.add(Dense(3, activation='softmax'))
model.compile(loss = 'categorical_crossentropy', optimizer='adam', metrics = ['accuracy'])
print(model.summary())
```

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_7 (Embedding)     (None, 51, 120)           60000

 spatial_dropout1d_7 (Spati  (None, 51, 120)           0
 alDropout1D)

 lstm_7 (LSTM)               (None, 704)               2323200

 dense_10 (Dense)            (None, 352)               248160

 dense_11 (Dense)            (None, 3)                 1059

=================================================================
Total params: 2632419 (10.04 MB)
Trainable params: 2632419 (10.04 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
```

Parametrización e inserción de datos de entrenamiento

```
[68] #Splitting the data into training and testing
     y=pd.get_dummies(data['Sentiment'])
     X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.3, random_state = 42)
```

Entrenamos el modelo

```
#Model Training
model.fit(X_train, y_train, epochs = 20, batch_size=32, verbose =1)
```

```
Epoch 1/20
129/129 [==============================] - 282s 2s/step - loss: 0.9511 - accuracy: 0.5488
Epoch 2/20
129/129 [==============================] - 274s 2s/step - loss: 0.8068 - accuracy: 0.6407
Epoch 3/20
129/129 [==============================] - 278s 2s/step - loss: 0.7265 - accuracy: 0.6807
Epoch 4/20
129/129 [==============================] - 277s 2s/step - loss: 0.6844 - accuracy: 0.6914
Epoch 5/20
129/129 [==============================] - 278s 2s/step - loss: 0.6389 - accuracy: 0.7154
Epoch 6/20
129/129 [==============================] - 280s 2s/step - loss: 0.6215 - accuracy: 0.7188
Epoch 7/20
129/129 [==============================] - 280s 2s/step - loss: 0.5961 - accuracy: 0.7338
Epoch 8/20
129/129 [==============================] - 279s 2s/step - loss: 0.5754 - accuracy: 0.7350
Epoch 9/20
129/129 [==============================] - 276s 2s/step - loss: 0.5370 - accuracy: 0.7457
Epoch 10/20
129/129 [==============================] - 276s 2s/step - loss: 0.5189 - accuracy: 0.7600
Epoch 11/20
129/129 [==============================] - 274s 2s/step - loss: 0.4988 - accuracy: 0.7590
Epoch 12/20
129/129 [==============================] - 278s 2s/step - loss: 0.4857 - accuracy: 0.7770
Epoch 13/20
129/129 [==============================] - 278s 2s/step - loss: 0.4585 - accuracy: 0.7811
Epoch 14/20
129/129 [==============================] - 276s 2s/step - loss: 0.4420 - accuracy: 0.7915
Epoch 15/20
129/129 [==============================] - 276s 2s/step - loss: 0.4368 - accuracy: 0.7910
Epoch 16/20
129/129 [==============================] - 277s 2s/step - loss: 0.4172 - accuracy: 0.8087
Epoch 17/20
129/129 [==============================] - 279s 2s/step - loss: 0.4112 - accuracy: 0.8027
Epoch 18/20
129/129 [==============================] - 276s 2s/step - loss: 0.3889 - accuracy: 0.8078
Epoch 19/20
129/129 [==============================] - 276s 2s/step - loss: 0.3841 - accuracy: 0.8131
Epoch 20/20
129/129 [==============================] - 278s 2s/step - loss: 0.3690 - accuracy: 0.8242
<keras.src.callbacks.History at 0x7e567eb3bc10>
```

Resultado

```
#Model Testing
model.evaluate(X_test,y_test)
```

```
56/56 [==============================] - 20s 357ms/step - loss: 1.2568 - accuracy: 0.6278
[1.2568222284317017, 0.627828061580658]
```

Usando el modelo Transformer-Based

Instalamos Transformers

```
!pip install transformers
✓ 5.0s

Requirement already satisfied: transformers in c:\users\néstor\anaconda3\lib\site-packages (4.32.1)
Requirement already satisfied: filelock in c:\users\néstor\anaconda3\lib\site-packages (from transformers) (3.9.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.15.1 in c:\users\néstor\anaconda3\lib\site-packages (from transformers) (0.15.1)
Requirement already satisfied: numpy>=1.17 in c:\users\néstor\anaconda3\lib\site-packages (from transformers) (1.24.3)
Requirement already satisfied: packaging>=20.0 in c:\users\néstor\anaconda3\lib\site-packages (from transformers) (23.1)
Requirement already satisfied: pyyaml>=5.1 in c:\users\néstor\anaconda3\lib\site-packages (from transformers) (6.0)
Requirement already satisfied: regex!=2019.12.17 in c:\users\néstor\anaconda3\lib\site-packages (from transformers) (2022.7.9)
Requirement already satisfied: requests in c:\users\néstor\anaconda3\lib\site-packages (from transformers) (2.31.0)
Requirement already satisfied: tokenizers!=0.11.3,<0.14,>=0.11.1 in c:\users\néstor\anaconda3\lib\site-packages (from transformers) (0.13.2)
Requirement already satisfied: safetensors>=0.3.1 in c:\users\néstor\anaconda3\lib\site-packages (from transformers) (0.3.2)
Requirement already satisfied: tqdm>=4.27 in c:\users\néstor\anaconda3\lib\site-packages (from transformers) (4.65.0)
Requirement already satisfied: fsspec in c:\users\néstor\anaconda3\lib\site-packages (from huggingface-hub<1.0,>=0.15.1->transformers) (2023.4.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in c:\users\néstor\anaconda3\lib\site-packages (from huggingface-hub<1.0,>=0.15.1->transformers) (4.7.1)
Requirement already satisfied: colorama in c:\users\néstor\anaconda3\lib\site-packages (from tqdm>=4.27->transformers) (0.4.6)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\néstor\anaconda3\lib\site-packages (from requests->transformers) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\néstor\anaconda3\lib\site-packages (from requests->transformers) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\néstor\anaconda3\lib\site-packages (from requests->transformers) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\néstor\anaconda3\lib\site-packages (from requests->transformers) (2023.7.22)
```

Importamos transformers y definimos el dataSet

```python
import transformers
from transformers import pipeline
sentiment_pipeline = pipeline("sentiment-analysis")
data = [
    "It was the best of times.",
    "t was the worst of times.",
    "Despite market challenges, XYZ Corporation is poised for growth and increased profitability.",
    "TechInnovate's groundbreaking technology has revolutionized the industry, earning widespread acclaim.",
    "A series of unfortunate events led to a decline in $ABC's market share and investor confidence.",
    "XYZ Pharma's commitment to research and development resulted in a breakthrough drug, positively impacting its stock value.",
    "Global economic uncertainties and trade tensions create a negative outlook for investors.",
    "In a strategic move, $XYZ acquires a key competitor, signaling potential market dominance.",
    "XYZ Airlines faces operational challenges, leading to a decline in customer satisfaction and stock value.",
    "In response to consumer demand, XYZ Retailers introduces sustainable practices, earning positive reviews.",
    "Despite concerns about inflation, $ABC Financials reports strong quarterly earnings, exceeding expectations.",
    "In a surprising turn, $XYZ's innovative approach to cybersecurity earns positive recognition from industry experts.",
    "The unexpected departure of key executives at XYZ Inc. results in uncertainty and negative market sentiment.",
    "Positive economic indicators and strong corporate earnings contribute to a bullish market outlook.",
    "XYZ Biotech faces regulatory hurdles in the approval process for a crucial drug, causing a temporary setback.",
    "Amidst industry disruptions, $ABC emerges as a frontrunner, reporting positive quarterly results.",
    "Rising interest rates and inflation concerns contribute to a negative outlook for the real estate sector.",
    "TechPioneer's commitment to diversity and inclusion initiatives receives positive recognition.",
    "In a bold move, XYZ Energy Corp. invests $200 million in renewable energy projects, earning positive reviews.",
    "Despite a temporary dip in stock prices, $ABC's long-term growth prospects have investors feeling optimistic.",
    "ABC Inc. faces continued financial challenges, leading to concerns among stakeholders about its stability.",
    "GlobalTrade Expo witnesses a significant increase in participation, reflecting a positive outlook on international commerce."
    "Market analysts predict a stable quarter for $XYZ, with a neutral outlook on the company's performance.",
    "GreenTech Solutions announces a major breakthrough in renewable energy research.",
    "Economic uncertainties and global tensions contribute to a bearish trend in the stock market.",
    "In response to changing consumer preferences, XYZ Fashion rebrands and introduces sustainable fashion lines.",
    "XYZ Motors faces production delays, leading to a decrease in stock value and concerns among shareholders.",
    "Despite challenges in the automotive industry, XYZ Motors reports a steady increase in sales.",
    "Amidst geopolitical tensions, the TechSummit attracts top industry leaders, fostering discussions on global technological col
    "After a successful merger, XYZ PharmaCorp streamlines operations, resulting in a 15% increase in efficiency.",
    "After months of speculation, BlueChip Ventures successfully exits its investment, showcasing strategic financial planning.",
    "Despite economic challenges, XYZ Construction reports a steady growth in revenue.",
    "In a surprising twist, XYZ Auto announces plans to shift production to electric vehicles.",
    "Despite a challenging economic climate, $ABC announces strong Q3 earnings, defying expectations.",
    "An unexpected legal victory for $XYZ boosts investor confidence, leading to a positive market response and an increase in sto
    "Unfavorable exchange rates and geopolitical tensions contribute to a negative impact on international trade.",
    "TechInnovate's CEO resignation sends shockwaves through the industry, impacting the company's stock negatively.",
    "Market analysts remain cautiously optimistic as $XYZ announces plans for expansion into emerging markets.",
    "In a groundbreaking announcement, TechInnovate unveils plans for a $500 million research and development facility.",
    "XYZ Aerospace secures a lucrative government contract for the development of advanced drone technology.",
```

Insertamos el dataSet en el modelo y nos muestra el resultado

```
sentiment_pipeline(data)
✓ 6.7s
No model was supplied, defaulted to distilbert-base-uncased-finetuned-sst-2-english and revision af0f99b (https://huggingface.co/distilbert-base-uncased-finetuned-sst-2-english).
Using a pipeline without specifying a model name and revision in production is not recommended.
All PyTorch model weights were used when initializing TFDistilBertForSequenceClassification.

All the weights of TFDistilBertForSequenceClassification were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFDistilBertForSequenceClassification for predictions without further training.

[{'label': 'POSITIVE', 'score': 0.9994569420814514},
 {'label': 'NEGATIVE', 'score': 0.9987302422523499},
 {'label': 'POSITIVE', 'score': 0.9995840191841125},
 {'label': 'POSITIVE', 'score': 0.9998432397842407},
 {'label': 'NEGATIVE', 'score': 0.9996845722198486},
 {'label': 'POSITIVE', 'score': 0.9996333122253418},
 {'label': 'NEGATIVE', 'score': 0.9989424347877502},
 {'label': 'POSITIVE', 'score': 0.9974205493927002},
 {'label': 'NEGATIVE', 'score': 0.9996579885482788},
 {'label': 'POSITIVE', 'score': 0.9995850920677185},
 {'label': 'POSITIVE', 'score': 0.9992934465408325},
 {'label': 'POSITIVE', 'score': 0.9998249411582947},
 {'label': 'NEGATIVE', 'score': 0.9996548891067505},
 {'label': 'POSITIVE', 'score': 0.9988974332809448},
 {'label': 'NEGATIVE', 'score': 0.999160885810852},
 {'label': 'POSITIVE', 'score': 0.999308705329895},
 {'label': 'NEGATIVE', 'score': 0.9995489716529846},
 {'label': 'POSITIVE', 'score': 0.9997730851173401},
 {'label': 'POSITIVE', 'score': 0.999821126461029},
 {'label': 'POSITIVE', 'score': 0.9982261061668396},
 {'label': 'NEGATIVE', 'score': 0.9619324803352356},
 {'label': 'POSITIVE', 'score': 0.9998257756233215},
 {'label': 'NEGATIVE', 'score': 0.9962846040725708},
 {'label': 'POSITIVE', 'score': 0.9996789693832397},
 {'label': 'NEGATIVE', 'score': 0.9338471293449402},
...
```