COLOMBIA POTENCIA DE LA VIDA

TIC

# Actividad 2

## NLP sobre IMDB Dataset of 50K Movie Reviews

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

# NLP sobre IMDB Dataset of 50K Movie Reviews

Ejemplo de modelo de clasificación de texto para IMDB Dataset of 50K Movie Reviews

El conjunto de datos IMDB de 50,000 revisiones de películas es uno de los conjuntos de datos más utilizados en el campo del procesamiento del lenguaje natural (NLP). Este conjunto de datos se compone de 50,000 revisiones de películas en inglés extraídas de la base de datos de películas IMDB. Estas revisiones están etiquetadas como positivas o negativas, lo que significa que cada revisión está asociada con una etiqueta que indica si la opinión expresada es positiva o negativa con respecto a la película.

# Aspectos importantes sobre el conjunto de datos IMDB de 50,000 revisiones de películas:

**Tamaño:**
Como su nombre lo indica, el conjunto de datos consta de 50,000 revisiones de películas.

**División de Datos:**
El conjunto de datos está dividido en dos partes: un conjunto de entrenamiento y un conjunto de prueba. El conjunto de entrenamiento contiene 25,000 revisiones etiquetadas, mientras que el conjunto de prueba contiene las 25,000 revisiones restantes para evaluar el rendimiento de los modelos.

**Etiquetas:**
Cada revisión está asociada con una etiqueta de sentimiento que indica si la revisión es positiva o negativa. Las etiquetas están equilibradas en el conjunto de datos, lo que significa que hay aproximadamente la misma cantidad de revisiones positivas y negativas.

**Formato de Datos:**
Cada revisión es una secuencia de palabras, donde las palabras están tokenizadas y se representa un índice único para cada palabra. Esto permite que las revisiones se procesen fácilmente utilizando técnicas de aprendizaje profundo.

**Uso en Investigación y Práctica:**
Debido a su tamaño razonable y a la calidad de las etiquetas proporcionadas, el conjunto de datos IMDB de 50,000 revisiones de películas es ampliamente utilizado en la investigación y la práctica en el campo de NLP. Se utiliza para entrenar y evaluar modelos de clasificación de texto, como modelos de redes neuronales recurrentes (RNN), modelos de redes neuronales convolucionales (CNN), modelos de atención y transformadores, entre otros.

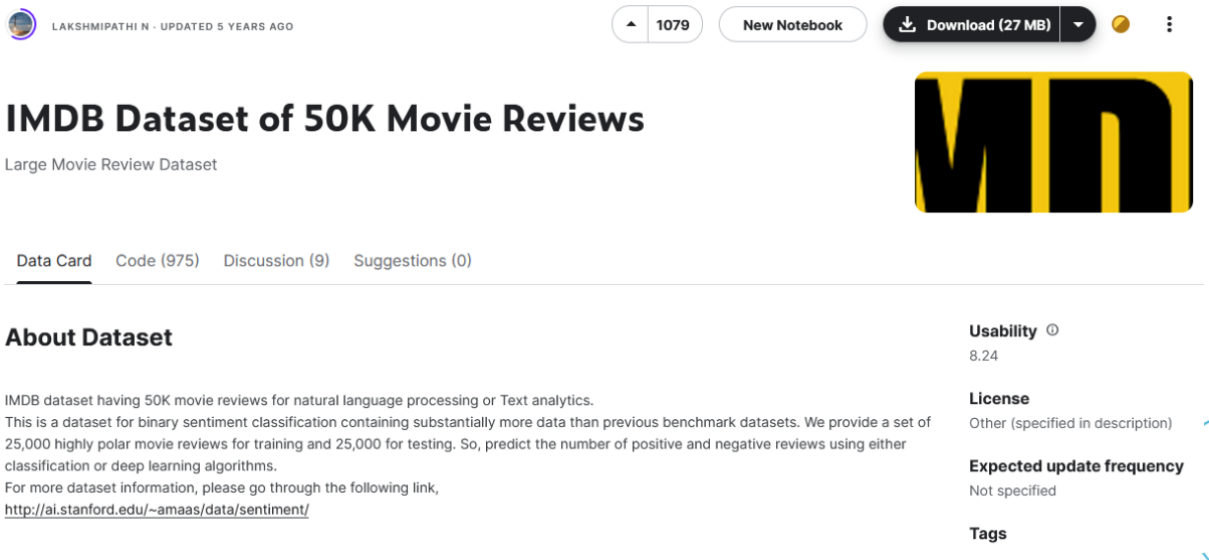
El conjunto de datos IMDB de 50,000 revisiones de películas es una herramienta invaluable para el desarrollo y la evaluación de modelos de análisis de sentimientos y clasificación de texto en el ámbito del procesamiento del lenguaje natural.

# Descarga del conjunto de datos

Esta base da datos la podemos descargar de kagle en el siguiente enlace.

**Enlace**

Entrando al enlace vemos la opción de descarga:




La descarga nos entrega un comprimido con un archivo csv dentro:

# Explorando el IMDB Dataset of 50K Movie Reviews

Este archivo CSV lo debemos llevar a nuestro entorno de trabajo y ahora podemos empezar a explorarlo. Ahora vamos a realizar varias operaciones para explorar y comprender el conjunto de datos que contiene revisiones de películas etiquetadas con sentimientos positivos o negativos.
Importación de bibliotecas:

El primer paso es importar las bibliotecas necesarias. En este caso, se utiliza la biblioteca Pandas para trabajar con estructuras de datos tabulares, como DataFrames en Python.

```
import pandas as pd
```

# Explorando el IMDB Dataset of 50K Movie Reviews

**Importación del conjunto de datos:**

El conjunto de datos "IMDB Dataset.csv" se carga en un DataFrame de Pandas llamado df. Esto se realiza utilizando la función pd.read_csv(), que lee el archivo CSV y crea un DataFrame con los datos.

```
#importing dataset
df=pd.read_csv("IMDB Dataset.csv")
print(df.head())
```

**Visualización de las primeras filas del conjunto de datos:**

Se imprime una vista previa de las primeras filas del DataFrame df utilizando el método head(). Esto proporciona una idea inicial de la estructura y el contenido del conjunto de datos.

```
                                              review sentiment
0  One of the other reviewers has mentioned that ...  positive
1  A wonderful little production. <br /><br />The...  positive
2  I thought this was a wonderful way to spend ti...  positive
3  Basically there's a family where a little boy ...  negative
4  Petter Mattei's "Love in the Time of Money" is...  positive
```

# Explorando el IMDB Dataset of 50K Movie Reviews

**Descripción del conjunto de datos:**

Se utiliza el método describe() para generar estadísticas descriptivas del conjunto de datos, como el recuento, la media, la desviación estándar, el mínimo y el máximo de cada columna numérica en el DataFrame. Esto ayuda a comprender la distribución y las características de las variables numéricas.

```
#Checking Dataset Description
print(df.describe())
```

**Recuento de sentimientos:**

Se calcula el recuento de valores únicos en la columna 'sentiment' para determinar cuántas revisiones están etiquetadas como positivas y cuántas como negativas. Esto proporciona información sobre el equilibrio de clases en el conjunto de datos.

```
#sentiment count
print("Total :\n",df['sentiment'].value_counts())
```

```
Total :
 positive    25000
negative    25000
```

# Explorando el IMDB Dataset of 50K Movie Reviews

*Este código carga un conjunto de datos de revisiones de películas, muestra las primeras filas del conjunto de datos, proporciona estadísticas descriptivas y cuenta el número de revisiones etiquetadas como positivas y negativas para comprender mejor la naturaleza del conjunto de datos.*

# Preprocesamiento de las secuencias

Ahora se realizan varias operaciones para preparar los datos de revisión de películas para su posterior procesamiento y modelado.

**Inicialización de listas:**
Se crea una lista vacía llamada review para almacenar las revisiones de películas y otra lista llamada sentences que contiene todas las revisiones en forma de oraciones. La lista sentences se obtiene convirtiendo la columna 'review' del DataFrame IMDB en una lista.

**Bucle de iteración para almacenar revisiones:**
Se recorre cada revisión en la lista de oraciones sentences y se agrega cada revisión a la lista review. Esto se hace para convertir la lista de oraciones en una lista de revisiones completas.

```
review = []
sentences = list(IMDB['review'])
for sen in sentences:
    review.append(sen)
```

# Preprocesamiento de las secuencias

**Creación de etiquetas:**

Se extraen las etiquetas de sentimiento ('positive' o 'negative') del DataFrame IMDB y se convierten en un arreglo NumPy de etiquetas binarias, donde 'positive' se convierte en 1 y 'negative' se convierte en 0. Esto se hace utilizando la función map() y una función lambda.

```
labels = IMDB['sentiment']
labels = np.array(list(map(lambda x: 1 if x=="positive" else 0, labels)))
```

**División de datos de entrenamiento y prueba:**

Se utilizan los datos de revisiones y etiquetas preparados para dividir el conjunto de datos en datos de entrenamiento y prueba. Esto se hace utilizando la función train_test_split() de Scikit-learn, que divide aleatoriamente los datos en un conjunto de entrenamiento y un conjunto de prueba, con un tamaño de prueba del 20%.

```
from sklearn.model_selection import train_test_split
train_sentences, test_sentences, train_labels, test_labels = train_test_split(review, labels, test_size=0.20)
```

# Preprocesamiento de las secuencias

```
from sklearn.model_selection import train_test_split
train_sentences, test_sentences, train_labels, test_labels =
train_test_split(review, labels, test_size=0.20)
```

Este código prepara los datos de revisión de películas para el modelado dividiéndolos en datos de entrenamiento y prueba, y convierte las etiquetas de sentimiento en una forma adecuada para el modelado.

# Tokenización y generación de secuencias

Se realiza la tokenización y el acolchado de las secuencias de texto para que puedan ser procesadas por un modelo de aprendizaje automático.

**Parámetros de configuración:**

Se definen varios parámetros para la tokenización y el acolchado de las secuencias de texto, como el tamaño del vocabulario (vocab_size), la longitud máxima de las secuencias (max_length), la dimensión del embedding (embedding_dim), el tipo de truncado (trunc_type) y el token para palabras fuera del vocabulario (oov_tok).

```
# Parameters
vocab_size = 1000
max_length = 120
embedding_dim = 16
trunc_type='post'
oov_tok = "<OOV>"
```

# Tokenización y generación de secuencias

**Tokenización y acolchado de secuencias:**

Se importan las clases Tokenizer y pad_sequences de Keras para realizar la tokenización y el acolchado de las secuencias de texto.

Se inicializa un objeto Tokenizer con el parámetro num_words establecido en el tamaño del vocabulario y el token OOV especificado.

Se ajusta el tokenizador a las frases de entrenamiento (train_sentences) para construir el índice de palabras.

Se generan secuencias de índices para las frases de entrenamiento y prueba utilizando el método texts_to_sequences() del tokenizador.

Las secuencias se acolchan con ceros o se truncatan según la longitud máxima especificada utilizando la función pad_sequences().

# Tokenización y generación de secuencias

**Tokenización y acolchado de secuencias:**

Este código tokeniza y acolcha las secuencias de texto para que tengan una longitud uniforme y puedan ser utilizadas como entrada para un modelo de aprendizaje automático.

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences

# Initialize the Tokenizer class
tokenizer = Tokenizer(num_words = vocab_size, oov_token=oov_tok)

# Generate the word index dictionary for the training sentences
tokenizer.fit_on_texts(train_sentences)
word_index = tokenizer.word_index

# Generate and pad the training sequences
sequences = tokenizer.texts_to_sequences(train_sentences)
padded = pad_sequences(sequences,maxlen=max_length, truncating=trunc_type)

# Generate and pad the test sequences
test_sequences = tokenizer.texts_to_sequences(test_sentences)
test_padded = pad_sequences(test_sequences,maxlen=max_length, truncating=trunc_type)
```

# Construcción y entrenamiento del modelo

Se construye y entrena un modelo de red neuronal para la clasificación binaria de secuencias de texto.

**Construcción del modelo:**

Se crea un modelo secuencial (Sequential) de Keras.
La primera capa es una capa de embedding (Embedding), que convierte los índices de palabras en vectores densos de embedding. El tamaño del vocabulario, la dimensión del embedding y la longitud máxima de entrada se especifican como parámetros.
La capa de embedding está seguida por una capa de aplanamiento (Flatten), que convierte los datos en un formato adecuado para la entrada en una capa densa.
Se agrega una capa densa (Dense) con 64 unidades y activación ReLU.
Finalmente, se añade una capa densa con una sola unidad y activación sigmoide, que produce la salida binaria.

ENLACE

```
# Build the model
model = keras.Sequential([
    keras.layers.Embedding(vocab_size, embedding_dim, input_length=max_length),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])
```

# Construcción y entrenamiento del modelo

**Compilación del modelo:**

Se compila el modelo utilizando el optimizador 'adam' y la función de pérdida 'binary_crossentropy', ya que se trata de un problema de clasificación binaria.
Se especifica 'accuracy' como la métrica para evaluar el rendimiento del modelo.

**Resumen del modelo:**

Se imprime un resumen del modelo, que muestra la arquitectura de la red neuronal, el número de parámetros entrenables y el flujo de datos a través de las diferentes capas.

```
# Setup the training parameters
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

# Print the model summary
model.summary()
```

# Construcción y entrenamiento del modelo

**Entrenamiento del modelo:**

Se especifica el número de épocas (num_epochs) para el entrenamiento.

Se entrena el modelo utilizando los datos de entrenamiento (padded) y las etiquetas de entrenamiento (train_labels), y se valida con los datos de prueba (test_padded) y las etiquetas de prueba (test_labels).

Este código construye, compila y entrena un modelo de red neuronal para clasificar secuencias de texto en dos categorías. El modelo utiliza una capa de embedding seguida de capas densas, y se entrena utilizando el optimizador Adam y la función de pérdida de entropía cruzada binaria.

```
# Train the model
history = model.fit(padded,
                    train_labels,
                    epochs=num_epochs,
                    validation_data=(test_padded, test_labels))

# Graficar el historial de entrenamiento:
pd.DataFrame(history.history).plot(grid=True)
```

# *Preguntas:*

1. ¿Qué hace el código en general?
2. ¿Cuál es el propósito de cada parte del código?
3. ¿Por qué se utilizan los parámetros específicos en cada función (por ejemplo, num_words, embedding_dim, input_length, loss, optimizer, etc.)?
4. ¿Cuál es la diferencia entre train_sentences y test_sentences?
5. ¿Qué significan vocab_size, max_length, embedding_dim y oov_tok?
6. ¿Por qué se utilizan las capas Embedding, Flatten, Dense y sigmoid en el modelo?
7. ¿Qué representan las métricas de pérdida y precisión en el modelo?
8. ¿Qué pasa durante el entrenamiento del modelo?
9. ¿Qué significan las épocas y cómo afectan al proceso de entrenamiento?
10. ¿Por qué es importante la validación del modelo utilizando los datos de prueba?
11. ¿Cuál es el objetivo final del código?

# *Ejercicios:*

1. Modifica los parámetros del modelo (como el tamaño del vocabulario, la longitud máxima, la dimensión de incrustación, etc.) y observa cómo afecta al rendimiento del modelo.
2. Experimenta con diferentes arquitecturas de modelos (agregando o eliminando capas, cambiando las funciones de activación, etc.) y observa cómo afecta al rendimiento.
3. Intenta utilizar un algoritmo de optimización diferente y observa cómo afecta al rendimiento del modelo.
4. Divide los datos en diferentes proporciones para entrenamiento y prueba y observa cómo afecta al rendimiento del modelo.
5. Explora otras métricas de evaluación del modelo y compara su rendimiento con las métricas existentes.

Estos ejercicios te ayudarán a comprender mejor cómo funciona el código y a familiarizarte con los conceptos y técnicas relacionados con el procesamiento de texto y la creación de modelos de redes neuronales.

COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira