COLOMBIA POTENCIA DE LA VIDA

TIC

Actividad 1

Introducción keras en NLP

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Introducción keras en NLP

Para empezar vamos a ver el Tokenizer de keras.preprocessing.text es una herramienta que se utiliza para tokenizar texto, es decir, para convertir secuencias de texto en secuencias de números enteros.

El Tokenizer es una clase en la biblioteca Keras que se utiliza para procesar texto y convertirlo en una forma numérica que pueda ser entendida y procesada por modelos de redes neuronales. Es una herramienta fundamental en tareas de procesamiento del lenguaje natural, donde la representación del texto es un paso crucial para el éxito del modelo.

Las funcionalidades principales del Tokenizer son

Tokenización:

Una de las funciones clave del Tokenizer es la tokenización, que implica dividir el texto en tokens o palabras individuales. Este proceso es esencial ya que permite al modelo comprender y procesar el significado de las palabras en el texto. La tokenización también puede incluir la eliminación de signos de puntuación, caracteres especiales y conversiones a minúsculas para estandarizar el texto y facilitar su procesamiento.

Indexación:

Una vez que el texto se ha tokenizado, el Tokenizer asigna un índice único a cada palabra en el vocabulario, basado en la frecuencia de aparición de cada palabra en el corpus de texto. Esta indexación es importante porque convierte las palabras en representaciones numéricas que pueden ser interpretadas por el modelo de red neuronal.

Las funcionalidades principales del Tokenizer son

Vectorización:

Otra funcionalidad del Tokenizer es la vectorización, donde las secuencias de palabras se convierten en secuencias de números enteros, utilizando los índices asignados durante la indexación. Este paso es esencial para alimentar los datos al modelo de red neuronal, ya que las redes neuronales requieren entradas numéricas para realizar cálculos y aprender patrones en los datos.

Padding:

Además, el Tokenizer ofrece opciones para el relleno (padding) y el truncamiento de las secuencias de palabras. El relleno implica agregar ceros al principio o al final de las secuencias para que todas tengan la misma longitud, lo que permite procesar lotes de datos de manera eficiente. Por otro lado, el truncamiento limita la longitud de las secuencias para unificar su tamaño y mejorar la velocidad de procesamiento.

Las funcionalidades principales del Tokenizer son

El Tokenizer es una herramienta versátil y poderosa que facilita la preparación de datos en tareas de NLP al convertir texto en representaciones numéricas comprensibles para los modelos de redes neuronales. Su capacidad para tokenizar, indexar, vectorizar y manipular secuencias de palabras lo convierte en un componente esencial en el flujo de trabajo de procesamiento del lenguaje natural con Keras.

Ejemplo de Tokenización con keras

1. Utilizan la clase Tokenizer de la biblioteca Keras para procesar un conjunto de frases y realizar varias operaciones relacionadas con la tokenización y la secuenciación de texto.

Se importa el módulo "keras" que contiene la clase Tokenizer.

```
import keras
```

Se define una lista llamada "frases" que contiene tres cadenas de texto.

```
frases = [
    'Hola mundo',
    'Hola a todos',
    'Hola a todo el mundo'
]
```

Ejemplo de Tokenización con keras

Se crea una instancia de la clase Tokenizer con el parámetro "num_words" establecido en 10. Esto significa que solo se considerarán las 10 palabras más comunes en el conjunto de texto para la tokenización.
La instancia del Tokenizer se ajusta a las frases proporcionadas mediante el método "fit_on_texts", lo que genera un diccionario de tokens basado en las palabras presentes en las frases.

```
#Genera el diccionario de tokens
tokenizer = keras.preprocessing.text.Tokenizer(num_words = 10)
tokenizer.fit_on_texts(frases)
word_index = tokenizer.word_index
print('word_index =',word_index)
```

Luego se imprime el diccionario generado, que muestra la correspondencia entre las palabras y los tokens asignados.

```
word_index = {'hola': 1, 'mundo': 2, 'a': 3, 'todos': 4, 'todo': 5, 'el': 6}
```

Ejemplo de Tokenización con keras

Se llama al método "texts_to_sequences" de Tokenizer para convertir las frases en secuencias de tokens. Esto reemplaza cada palabra en las frases con su token correspondiente según el diccionario generado previamente.

```
#Generacion de secuencia tokenizadas
secuencias = tokenizer.texts_to_sequences(frases)
print('secuencias =',secuencias)
```

Se imprime la secuencia de tokens resultante para cada frase.

```
secuencias = [[1, 2], [1, 3, 4], [1, 3, 5, 6, 2]]
```

Aqui podemos observar que las secuencias tienen longitudes diferentes

Ejemplo de Tokenización con keras

Entonces se llama al método "pad_sequences" de la clase preprocessing.sequence para rellenar las secuencias de tokens a una longitud uniforme. Por defecto, este método rellena con ceros al principio de cada secuencia hasta alcanzar la longitud de la secuencia más larga.

```
#Rellena las secuencias a una longitud uniforme
relleno = keras.preprocessing.sequence.pad_sequences(secuencias)
print('rellena =\n',relleno)
```

Finalmente se imprime la matriz de secuencias rellenadas.

```
rellena =
 [[0 0 0 1 2]
 [0 0 1 3 4]
 [1 3 5 6 2]]
```

Este código muestra cómo usar el Tokenizer de Keras para tokenizar texto, generar secuencias de tokens y rellenar las secuencias para igualar su longitud. Esto es útil para preparar datos de texto para su uso en modelos de redes neuronales.

Ejemplo de Tokenización con keras

2. Ahora vamos a realizar algunos cambios en el código para observar los resultados obtenidos.

2.1. Que pasa si el número de palabra en las frase es mayor al parámetro (num_words)

```
frases = [
    'Hola mundo',
    'Hola a todos',
    'Hola a todo el mundo',
    'Buen dia, como estas hoy'
]

#Genera el diccionario de tokens
tokenizer = keras.preprocessing.text.Tokenizer(num_words = 10)
tokenizer.fit_on_texts(frases)
word_index = tokenizer.word_index
print('\nword_index =',word_index)

#Generacion de secuencia tokenizadas
secuencias = tokenizer.texts_to_sequences(frases)
print('secuencias =',secuencias)

#Rellena las secuencias a una longitud uniforme
relleno = keras.preprocessing.sequence.pad_sequences(secuencias)
print('rellena =\n',relleno)
```

Ejemplo de Tokenización con keras

Los resultados serían:

word_index = {'hola': 1, 'mundo': 2, 'a': 3, 'todos': 4, 'todo': 5, 'el': 6, 'buen': 7, 'dia': 8, 'como': 9, 'estas': 10, 'hoy': 11}

secuencias = [[1, 2], [1, 3, 4], [1, 3, 5, 6, 2], [7, 8, 9]]

rellena =

[[0 0 0 1 2]
[0 0 1 3 4]
[1 3 5 6 2]
[0 0 7 8 9]]

Discute los cambios en los resultados

Ejemplo de Tokenización con keras

2.2. Que pasa si el numero de palabra en las frase es mayor al parametro (num_words) ademas incluimos el parametro OOV

El parámetro oov_token (out of vocabulary token)
En el Tokenizer de Keras se utiliza para especificar un token que se asignará a las palabras fuera del vocabulario durante el proceso de tokenización. Es decir, si durante la tokenización se encuentra una palabra que no está en el diccionario de tokens creado previamente, esta palabra se reemplazará por el token especificado en el parámetro oov_token.

Ejemplo de Tokenización con keras

Por ejemplo, si se define oov_token='<OOV>', entonces todas las palabras desconocidas se representarán con el token '<OOV>'. Esto puede ser útil para manejar palabras desconocidas o fuera del vocabulario de manera consistente durante el procesamiento de texto.

```
frases = [
    'Hola mundo',
    'Hola a todos',
    'Hola a todo el mundo',
    'Buen dia, como estas hoy'
]

#Genera el diccionario de tokens
tokenizer = keras.preprocessing.text.Tokenizer(num_words = 10,
                                               oov_token="<OOV>")
tokenizer.fit_on_texts(frases)
word_index = tokenizer.word_index
print('\nword_index =',word_index)

#Generacion de secuencia tokenizadas
secuencias = tokenizer.texts_to_sequences(frases)
print('secuencias =',secuencias)

#Rellena las secuencias a una longitud uniforme
relleno = keras.preprocessing.sequence.pad_sequences(secuencias)
print('rellena =\n',relleno)
```

Ejemplo de Tokenización con keras

Los resultados serian:

word_index = {'<OOV>': 1, 'hola': 2, 'mundo': 3, 'a': 4, 'todos': 5, 'todo': 6, 'el': 7, 'buen': 8, 'dia': 9, 'como': 10, 'estas': 11, 'hoy': 12}

secuencias = [[2, 3], [2, 4, 5], [2, 4, 6, 7, 3], [8, 9, 1, 1, 1]]

rellena =

[[0 0 0 2 3]
[0 0 2 4 5]
[2 4 6 7 3]
[8 9 1 1 1]]

Ejemplo de Tokenización con keras

3. Ahora vamos a probar los parámetros padding y truncating de la función pad_sequences en Keras se utilizan para controlar el relleno y el truncamiento de las secuencias durante el preprocesamiento de datos.

Padding (Relleno):

El parámetro padding determina cómo se rellenan las secuencias que son más cortas que la longitud máxima especificada. Puede tomar dos valores:

- 'pre': Rellena antes de la secuencia, es decir, agrega los valores de relleno al principio de la secuencia.
- 'post': Rellena después de la secuencia, es decir, agrega los valores de relleno al final de la secuencia.

Ejemplo de Tokenización con keras

Truncating (Truncamiento):

El parámetro truncating determina cómo se truncan las secuencias que son más largas que la longitud máxima especificada. Puede tomar dos valores:

- 'pre': Trunca la secuencia desde el principio, es decir, elimina los elementos de la secuencia al principio.
- 'post': Trunca la secuencia desde el final, es decir, elimina los elementos de la secuencia al final.

Ejemplo de Tokenización con keras

Estos parámetros son útiles cuando se trabaja con secuencias de longitud variable y se necesita estandarizar su longitud para entrenar modelos de aprendizaje profundo. Por ejemplo, en tareas de procesamiento de texto como el análisis de sentimientos o la clasificación de texto, es común que las secuencias de palabras tengan longitudes diferentes. Al especificar valores para padding y truncating, podemos asegurarnos de que todas las secuencias tengan la misma longitud, lo que facilita el procesamiento y el entrenamiento del modelo.

```
import keras

frases = [
    'Hola mundo',
    'Hola a todos',
    'Hola a todo el mundo',
    'Buen dia, como estas hoy'
]

#Genera el diccionario de tokens
tokenizer = keras.preprocessing.text.Tokenizer(num_words = 10,
                                                oov_token="<OOV>")
tokenizer.fit_on_texts(frases)
word_index = tokenizer.word_index
print('\nword_index =',word_index)

#Generacion de secuencia tokenizadas
secuencias = tokenizer.texts_to_sequences(frases)
print('secuencias =',secuencias)

#Rellena las secuencias a una longitud uniforme
relleno = keras.preprocessing.sequence.pad_sequences(secuencias,
                                                     padding = 'post',
                                                     truncating =  'post')

print('rellena =\n',relleno)
```

Ejemplo de Tokenización con keras

Las salidas generadas serían:

word_index = {'<OOV>': 1, 'hola': 2, 'mundo': 3, 'a': 4, 'todos': 5, 'todo': 6, 'el': 7, 'buen': 8, 'dia': 9, 'como': 10, 'estas': 11, 'hoy': 12}

secuencias = [[2, 3], [2, 4, 5], [2, 4, 6, 7, 3], [8, 9, 1, 1, 1]]

rellena =

[[2 3 0 0 0]
[2 4 5 0 0]
[2 4 6 7 3]
[8 9 1 1 1]]

COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira