Introducción

Mientras escribo este artículo, 1,907,223,370 sitios web están activos en Internet y 2,722,460 correos electrónicos son enviados por segundo. Esta es una cantidad increíblemente grande de datos. Es imposible para un usuario obtener información de volúmenes tan grandes de datos. Además, una gran parte de estos datos es redundante o no contiene mucha información útil. La forma más eficiente de obtener acceso a las partes más importantes de los datos, sin tener que revisar datos redundantes e insignificantes, es resumir los datos de manera que contenga información no redundante y útil únicamente. Los datos pueden estar en cualquier forma, como audio, video, imágenes y texto. En este artículo, veremos cómo podemos usar técnicas de resumen de texto automático para resumir datos de texto.

El resumen de texto es un subdominio de [Procesamiento de lenguaje natural](https://stackabuse.com/what-is-natural-language-processing/) (PNL) que se ocupa de extraer resúmenes de enormes trozos de textos. Hay dos tipos principales de técnicas utilizadas para resumir textos: las técnicas basadas en PNL y las técnicas basadas en el aprendizaje profundo. En este artículo, veremos una técnica simple basada en PNL para el resumen de texto. No utilizaremos ninguna biblioteca de aprendizaje automático en este artículo. Más bien, simplemente usaremos [la biblioteca NLTK de Python](https://www.nltk.org/)para resumir los artículos de Wikipedia.

Pasos de resumen de texto

Explicaré los pasos involucrados en el resumen de texto utilizando técnicas de PNL con la ayuda de un ejemplo.

El siguiente es un párrafo de uno de los famosos discursos de Denzel Washington en la 48ª edición de los Premios de Imagen NAACP:

*Entonces, sigue trabajando. Sigue esforzándote. Nunca te rindas. Cáete siete veces, levántate ocho. La facilidad es una mayor amenaza para el progreso que las dificultades. La facilidad es una mayor amenaza para el progreso que las dificultades. Entonces, sigue moviéndote, sigue creciendo, sigue aprendiendo. Nos vemos en el trabajo*

Podemos ver en el párrafo anterior que básicamente está motivando a otros para que trabajen duro y nunca se rindan. Para resumir el párrafo anterior utilizando técnicas basadas en NLP, debemos seguir una serie de pasos, que se describirán en las siguientes secciones.

Convertir párrafos a oraciones

Primero necesitamos convertir todo el párrafo en oraciones. La forma más común de convertir párrafos en oraciones es dividir el párrafo cada vez que se encuentra un período. Entonces, si dividimos el párrafo en discusión en oraciones, obtenemos las siguientes oraciones:

1. Así que sigue trabajando
2. Sigue esforzándote
3. Nunca te rindas
4. Cáete siete veces, levántate ocho
5. La facilidad es una mayor amenaza para el progreso que las dificultades
6. La facilidad es una mayor amenaza para el progreso que las dificultades
7. Así que sigue moviéndote, sigue creciendo, sigue aprendiendo.
8. Nos vemos en el trabajo

Preprocesamiento de texto

Después de convertir el párrafo en oraciones, debemos eliminar todos los caracteres especiales, detener palabras y números de todas las oraciones. Después del preprocesamiento, obtenemos las siguientes frases:

1. sigue trabajando
2. sigue esforzándote
3. nunca dar
4. caer siete veces conseguir ocho
5. aliviar la amenaza mayor progreso dificultades
6. aliviar la amenaza mayor progreso dificultades
7. sigue moviendo sigue creciendo sigue aprendiendo
8. ver trabajo

Tokenizando las oraciones

Necesitamos tokenizar todas las oraciones para obtener todas las palabras que existen en las oraciones. Después de tokenizing las oraciones, obtenemos una lista de las siguientes palabras:

['keep',

'working',

'keep',

'striving',

'never',

'give',

'fall',

'seven',

'time',

'get',

'eight',

'ease',

'greater',

'threat',

'progress',

'hardship',

'ease',

'greater',

'threat',

'progress',

'hardship',

'keep',

'moving',

'keep',

'growing',

'keep',

'learning',

'see',

'work']

Encontrar la frecuencia ponderada de ocurrencia

A continuación, necesitamos encontrar la frecuencia ponderada de las ocurrencias de todas las palabras. Podemos encontrar la frecuencia ponderada de cada palabra al dividir su frecuencia por la frecuencia de la palabra que aparece con mayor frecuencia. La siguiente tabla contiene las frecuencias ponderadas para cada palabra:

| Palabra | Frecuencia | Frecuencia ponderada |
| --- | --- | --- |
| facilitar | 2 | 0.40 |
| ocho | 1 | 0,20 |
| otoño | 1 | 0,20 |
| obtener | 1 | 0,20 |
| dar | 1 | 0,20 |
| mayor | 2 | 0.40 |
| creciente | 1 | 0,20 |
| privación | 2 | 0.40 |
| mantener | 5 | 1.00 |
| aprendizaje | 1 | 0,20 |
| emocionante | 1 | 0,20 |
| Nunca | 1 | 0,20 |
| Progreso | 2 | 0.40 |
| ver | 1 | 0,20 |
| Siete | 1 | 0,20 |
| esforzarse | 1 | 0,20 |
| amenaza | 2 | 0.40 |
| hora | 1 | 0,20 |
| trabajo | 1 | 0,20 |
| trabajando | 1 | 0,20 |

Dado que la palabra "mantener" tiene la frecuencia más alta de 5, por lo tanto, la frecuencia ponderada de todas las palabras se ha calculado al dividir su número de ocurrencias por 5.

Reemplazar palabras por frecuencia ponderada en oraciones originales

El paso final es conectar la frecuencia ponderada en lugar de las palabras correspondientes en las oraciones originales y encontrar su suma. Es importante mencionar que la frecuencia ponderada de las palabras eliminadas durante el preprocesamiento (palabras de detención, puntuación, dígitos, etc.) será cero y, por lo tanto, no es necesario agregarlas, como se menciona a continuación:

| Frase | Suma de frecuencias ponderadas |
| --- | --- |
| Así que sigue trabajando | 1 + 0.20 = 1.20 |
| Sigue esforzándote | 1 + 0.20 = 1.20 |
| Nunca te rindas | 0.20 + 0.20 = 0.40 |
| Cáete siete veces, levántate ocho | 0.20 + 0.20 + 0.20 + 0.20 + 0.20 = 1.0 |
| La facilidad es una mayor amenaza para el progreso que las dificultades | 0.40 + 0.40 + 0.40 + 0.40 + 0.40 = 2.0 |
| La facilidad es una mayor amenaza para el progreso que las dificultades | 0.40 + 0.40 + 0.40 + 0.40 + 0.40 = 2.0 |
| Así que sigue moviéndote, sigue creciendo, sigue aprendiendo. | 1 + 0.20 + 1 + 0.20 + 1 + 0.20 = 3.60 |
| Nos vemos en el trabajo | 0.20 + 0.20 = 0.40 |
|  |  |

Ordenar oraciones en orden descendente de la suma

El paso final es ordenar las oraciones en orden inverso de su suma. Las oraciones con las frecuencias más altas resumen el texto. Por ejemplo, mire la oración con la suma más alta de frecuencias ponderadas:

*Así que sigue moviéndote, sigue creciendo, sigue aprendiendo.*

Puedes juzgar fácilmente de qué se trata el párrafo. De manera similar, puede agregar la oración con la segunda suma más alta de frecuencias ponderadas para tener un resumen más informativo. Echa un vistazo a las siguientes oraciones:

*Entonces, sigue moviéndote, sigue creciendo, sigue aprendiendo. La facilidad es una mayor amenaza para el progreso que las dificultades.*

Estas dos oraciones dan un resumen bastante bueno de lo que se dijo en el párrafo.

Resumiendo artículos de Wikipedia

Ahora sabemos cómo funciona el proceso de resumen de texto usando una técnica de PNL muy simple. En esta sección, usaremos la biblioteca NLTK de Python para resumir un artículo de Wikipedia.

Obteniendo artículos de Wikipedia

Antes de poder resumir los artículos de Wikipedia, debemos obtenerlos de la web. Para ello utilizaremos un par de bibliotecas. La primera biblioteca que necesitamos descargar es la [hermosa sopa,](https://www.crummy.com/software/BeautifulSoup/bs4/doc/) que es una utilidad muy útil de Python para el rastreo web. Ejecute el siguiente comando en el símbolo del sistema para descargar la utilidad Beautiful Soup.

$ pip install beautifulsoup4

Otra biblioteca importante que necesitamos para analizar XML y HTML es la biblioteca [lxml](https://lxml.de/" \t "_blank) . Ejecute el siguiente comando en el símbolo del sistema para descargar lxml:

$ pip install lxml

Ahora permite algo de código de Python para raspar datos de la web. El artículo que vamos a rastrear es el artículo de Wikipedia sobre [Inteligencia Artificial](https://en.wikipedia.org/wiki/Artificial_intelligence) . Ejecuta el siguiente script:

import bs4 as bs

import urllib.request

import re

scraped\_data = urllib.request.urlopen('https://en.wikipedia.org/wiki/Artificial\_intelligence')

article = scraped\_data

parsed\_article = bs.BeautifulSoup(article,'lxml')

paragraphs = parsed\_article.find\_all('p')

article\_text = ""

for p in paragraphs:

article\_text += p.text

En el script anterior, primero importamos las bibliotecas importantes requeridas para raspar los datos de la web. Luego usamos la urlopenfunción de la urllib.requestutilidad para raspar los datos. A continuación, debemos llamar a la readfunción en el objeto devuelto por la urlopenfunción para poder leer los datos. Para analizar los datos, usamos BeautifulSoupobject y le pasamos el objeto de datos raspados, es decir, articley el lxmlanalizador.

En los artículos de Wikipedia, todo el texto del artículo está incluido dentro de las <p>etiquetas. Para recuperar el texto necesitamos llamar a la find\_allfunción en el objeto devuelto por BeautifulSoup. El nombre de la etiqueta se pasa como un parámetro a la función. La find\_allfunción devuelve todos los párrafos del artículo en forma de una lista. Todos los párrafos se han combinado para recrear el artículo.

Una vez que se raspa el artículo, tenemos que hacer un preprocesamiento.

Final del formulario

Preprocesamiento

El primer paso de preprocesamiento es eliminar las referencias del artículo. Wikipedia, las referencias están entre corchetes. La siguiente secuencia de comandos elimina los corchetes y reemplaza los espacios múltiples resultantes por un solo espacio. Echa un vistazo a la secuencia de comandos a continuación:

*# Removing Square Brackets and Extra Spaces*

article\_text = re.sub(r'\[[0-9]\*\]', ' ', article\_text)

article\_text = re.sub(r'\s+', ' ', article\_text)

El article\_textobjeto contiene texto sin corchetes. Sin embargo, no queremos eliminar nada más del artículo ya que este es el artículo original. No eliminaremos otros números, signos de puntuación ni caracteres especiales de este texto, ya que usaremos este texto para crear resúmenes y las frecuencias de palabras ponderadas se reemplazarán en este artículo.

Para limpiar el texto y calcular las frecuencias ponderadas, crearemos otro objeto. Echa un vistazo a la siguiente secuencia de comandos:

*# Removing special characters and digits*

formatted\_article\_text = re.sub('[^a-zA-Z]', ' ', article\_text )

formatted\_article\_text = re.sub(r'\s+', ' ', formatted\_article\_text)

Ahora tenemos dos objetos article\_text, que contiene el artículo original y el formatted\_article\_textque contiene el artículo con formato. Usaremos formatted\_article\_textpara crear histogramas de frecuencia ponderada para las palabras y reemplazaremos estas frecuencias ponderadas con las palabras en el article\_textobjeto.

Convertir texto a oraciones

En este punto hemos procesado previamente los datos. A continuación, tenemos que convertir el artículo en oraciones. Usaremos el article\_textobjeto para convertir el artículo en una frase, ya que contiene paradas completas. El formatted\_article\_textno contiene ninguna puntuación y, por lo tanto, no se puede convertir en oraciones utilizando la parada completa como parámetro.

El siguiente script realiza la tokenización de la oración:

sentence\_list = nltk.sent\_tokenize(article\_text)

Encontrar la frecuencia ponderada de ocurrencia

Para encontrar la frecuencia de aparición de cada palabra, usamos la formatted\_article\_textvariable. Utilizamos esta variable para encontrar la frecuencia de ocurrencia ya que no contiene puntuación, dígitos u otros caracteres especiales. Echa un vistazo a la siguiente secuencia de comandos:

stopwords = nltk.corpus.stopwords.words('english')

word\_frequencies = {}

for word in nltk.word\_tokenize(formatted\_article\_text):

if word not in stopwords:

if word not in word\_frequencies.keys():

word\_frequencies[word] = 1

else:

word\_frequencies[word] += 1

En el script anterior, primero almacenamos todas las palabras vacías en inglés de la nltkbiblioteca en una stopwordsvariable. A continuación, repasamos todas las oraciones y luego las palabras correspondientes para verificar primero si son palabras vacías. Si no, procedemos a verificar si las palabras existen en el word\_frequencydiccionario word\_frequencies, es decir , o no. Si se encuentra la palabra por primera vez, se agrega al diccionario como una clave y su valor se establece en 1. De lo contrario, si la palabra existe previamente en el diccionario, su valor simplemente se actualiza en 1.

Finalmente, para encontrar la frecuencia ponderada, simplemente podemos dividir el número de ocurrencias de todas las palabras por la frecuencia de la palabra que aparece con mayor frecuencia, como se muestra a continuación:

maximum\_frequncy = max(word\_frequencies.values())

for word in word\_frequencies.keys():

word\_frequencies[word] = (word\_frequencies[word]/maximum\_frequncy)

Calculando puntajes de oraciones

Ahora hemos calculado las frecuencias ponderadas para todas las palabras. Ahora es el momento de calcular los puntajes de cada oración agregando frecuencias ponderadas de las palabras que aparecen en esa oración en particular. La siguiente secuencia de comandos calcula las puntuaciones de la oración:

sentence\_scores = {}

for sent in sentence\_list:

for word in nltk.word\_tokenize(sent.lower()):

if word in word\_frequencies.keys():

if len(sent.split(' ')) < 30:

if sent not in sentence\_scores.keys():

sentence\_scores[sent] = word\_frequencies[word]

else:

sentence\_scores[sent] += word\_frequencies[word]

En el script anterior, primero creamos un sentence\_scoresdiccionario vacío . Las claves de este diccionario serán las oraciones mismas y los valores serán las puntuaciones correspondientes de las oraciones. A continuación, hacemos un bucle a través de cada oración en el sentence\_listy tokenize la oración en palabras.

A continuación, comprobamos si la palabra existe en el word\_frequenciesdiccionario. Esta verificación se realiza desde que creamos la sentence\_listlista desde el article\_textobjeto; por otro lado, las frecuencias de palabras se calcularon utilizando el formatted\_article\_textobjeto, que no contiene palabras de parada, números, etc.

No queremos oraciones muy largas en el resumen, por lo tanto, calculamos la puntuación solo para oraciones con menos de 30 palabras (aunque puede modificar este parámetro para su propio caso de uso). A continuación, verificamos si la frase existe en el sentence\_scoresdiccionario o no. Si la oración no existe, la agregamos al sentence\_scoresdiccionario como una clave y le asignamos la frecuencia ponderada de la primera palabra en la oración, como su valor. Por el contrario, si la oración existe en el diccionario, simplemente agregamos la frecuencia ponderada de la palabra al valor existente.

Obteniendo el resumen

Ahora tenemos el sentence\_scoresdiccionario que contiene oraciones con su puntuación correspondiente. Para resumir el artículo, podemos tomar las mejores frases con las puntuaciones más altas. La siguiente secuencia de comandos recupera las 7 oraciones principales y las imprime en la pantalla.

import heapq

summary\_sentences = heapq.nlargest(7, sentence\_scores, key=sentence\_scores.get)

summary = ' '.join(summary\_sentences)

print(summary)

En el guión anterior, usamos la heapqbiblioteca y llamamos a su nlargestfunción para recuperar las 7 oraciones principales con las puntuaciones más altas.

El resumen de salida se ve así:

*La inteligencia artificial (IA), a veces llamada inteligencia de la máquina, es la inteligencia demostrada por las máquinas, en contraste con la inteligencia natural mostrada por los humanos y otros animales. Se utilizan muchas herramientas en IA, incluidas versiones de búsqueda y optimización matemática, redes neuronales artificiales y métodos basados ​​en estadísticas, probabilidad y economía. Los problemas (o metas) tradicionales de la investigación de la IA incluyen el razonamiento, la representación del conocimiento, la planificación, el aprendizaje, el procesamiento del lenguaje natural, la percepción y la capacidad para mover y manipular objetos. Cuando el acceso a las computadoras digitales se hizo posible a mediados de la década de 1950, la investigación de AI comenzó a explorar la posibilidad de que la inteligencia humana pudiera reducirse a la manipulación de símbolos. Una propuesta para lidiar con esto es asegurar que la primera IA generalmente inteligente sea 'AI amigable', y luego será capaz de controlar las IA desarrolladas posteriormente. Hoy en día, la gran mayoría de los investigadores actuales de la IA trabajan en su lugar en aplicaciones de "IA estrecha" (como el diagnóstico médico o la navegación en automóviles). El aprendizaje automático, un concepto fundamental de la investigación en IA desde el inicio del campo, es el estudio de algoritmos informáticos que mejoran automáticamente a través de la experiencia.*

Recuerde, dado que los artículos de Wikipedia se actualizan con frecuencia, es posible que obtenga resultados diferentes según el momento de ejecución del script.

Conclusión

Este artículo explica el proceso de resumen de texto con la ayuda de la biblioteca Python NLTK. El proceso de raspar artículos usando la BeautifulSoapbiblioteca también se ha cubierto brevemente en el artículo. Le recomendaré que elimine cualquier otro artículo de Wikipedia y vea si puede obtener un buen resumen del artículo o no.