Quintana Martínez Erick
Tratamiento de lenguaje natural

Similitud coseno entre textos usando bigramas

Código

```
import numpy as np
from math import *
from nltk.corpus import stopwords

def construir_diccionario(texto1, texto2):
    vocabulario = {}
    i = 0

    for palabra in texto1:
        if palabra not in vocabulario:
            vocabulario[palabra] = i
            i += 1


    for palabra in texto2:
        if palabra not in vocabulario:
            vocabulario[palabra] = i
            i += 1

    return vocabulario

def frecuencias_de_terminos(texto1, texto2):
    vocabulario = construir_diccionario(texto1, texto2)

    a = np.zeros(len(vocabulario))
    b = np.zeros(len(vocabulario))

    for palabra in texto1:
        i = vocabulario[palabra]
        a[i] += 1

    for palabra in texto2:
        i = vocabulario[palabra]
        b[i] += 1

    return a, b

def raiz_cuadrada(x):
    return round(sqrt(sum([a * a for a in x])),4)

def similitudCoseno(x, y):
    numerador = sum(a * b for a, b in zip(x, y))
    denominador = raiz_cuadrada(x) * raiz_cuadrada(y)

    sim_cos = round(numerador/float(denominador),4)

    return sim_cos
```

```
texto1 = "La inteligencia artificial (IA) actual funciona bien cuando la aplicas a un ámbito muy concreto: desde jugar al ajedrez a hacer un diagnóstico médico. Pero el gran reto en el siglo XXI, consiste en alcanzar una inteligencia artificial general (AGI por sus siglas en inglés), que es aquella IA capaz de aprender tareas intelectuales como lo hacen los humanos. En este contexto, destaca GPT-3, un modelo de IA que permite generar lenguaje escrito. Es lo que se conoce como un modelo de lenguaje auto-regresivo, es decir, 'un algoritmo que permite crear la siguiente mejor palabra que seguiría a un texto dado', explica César de Pablo, científico de datos en BBVA Data &Analytics."

texto2 = "Desde Yo, robot la humanidad ha vivido aterrada con la idea de que los robots cobren vida y consciencia propia. Puede que la inteligencia artificial nos ponga un paso más cerca de ese día, y según algunos eso ya habría ocurrido en los cuarteles de una de los gigantes tecnológicos más importantes del siglo XXI: Google. El programa LaMDA, una IA especializada en conversaciones, pudo haber cobrado consciencia según uno de los ingenieros participantes en el proyecto. LaMDA (Language Model for Dialogue Applications, modelo de lenguaje para aplicaciones de diálogo en español) es un programa diseñado para tener conversaciones realistas con un ser humano prestando atención a 'un aparente sinnúmero de temas', como sucede normalmente en nuestras pláticas."

texto3 = "El francés Aspect, el estadounidense Clauser y el austriaco Zeilinger son tres pioneros de los revolucionarios mecanismos de la física cuántica. Han sido premiados por sus descubrimientos sobre el 'entrelazamiento cuántico', un fenómeno por el que dos partículas cuánticas están perfectamente correlacionadas, independientemente de la distancia que las separe, segñun ha anunciado el jurado del Nobel."

def minusculas(texto):
    return texto.lower()

def remover_puntuaciones(texto):
    punctuaciones = '''!()-[]{};:'"\,<>./?@#$%^&*_~'''

    for c in texto:
        if c in punctuaciones:
            texto = texto.replace(c, '')

    return texto

def tokenizar(texto):
    return texto.split(' ')

def generar_bigramas(tokens):
    bigramas=[]

    for i in range(len(tokens)-1):
        bigrama = [tokens[j] for j in range(i,i + 2)]
        bigramas.append(" ".join(bigrama))

    return bigramas

texto1 = minusculas(texto1)
texto1 = remover_puntuaciones(texto1)
texto1_tokenizado = tokenizar(texto1)
texto1_bigramas = generar_bigramas(texto1_tokenizado)
```

```
texto2 = minusculas(texto2)
texto2 = remover_puntuaciones(texto2)
texto2_tokenizado = tokenizar(texto2)
texto2_bigramas = generar_bigramas(texto2_tokenizado)

texto3 = minusculas(texto3)
texto3 = remover_puntuaciones(texto3)
texto3_tokenizado = tokenizar(texto3)
texto3_bigramas = generar_bigramas(texto3_tokenizado)

a, b = frecuencias_de_terminos(texto1_bigramas, texto2_bigramas)
print(f"Similitud entre texto 1 y texto 2 = {similitudCoseno(a, b)}")

a, b = frecuencias_de_terminos(texto1_bigramas, texto3_bigramas)
print(f"Similitud entre texto 1 y texto 3 = {similitudCoseno(a, b)}")

a, b = frecuencias_de_terminos(texto2_bigramas, texto3_bigramas)
print(f"Similitud entre texto 2 y texto 3 = {similitudCoseno(a, b)}")
```

Salida

```
Similitud entre texto 1 y texto 2 = 0.083
Similitud entre texto 1 y texto 3 = 0.012
Similitud entre texto 2 y texto 3 = 0.0242
```