Daniel Cu Sánchez
A01703613
Actividad final 2. Similitud en textos mediante Cadenas de Markov

Instrucciones

1.Implementa el algoritmo de similitud mediante diagramas de transición.

2.- Utiliza el ejemplo visto en clase para verificar su funcionamiento.

3.- Inventa otros dos textos y verifica su similitud.

Entregables: código fuente y PDF con los programas funcionando.

```
import numpy as np

# Función para calcular la matriz de transición de un texto
def calcular_matriz_transicion(texto):
  # Obtener el conjunto de caracteres únicos en el texto
  caracteres_unicos = list(set(texto))

  # Crear una matriz de transición de tamaño NxN, donde N es el número de caracteres únicos
  matriz_transicion = np.zeros((len(caracteres_unicos), len(caracteres_unicos)))

  # Calcular las frecuencias de transición de caracteres
  for i in range(len(texto) - 1):
    caracter_actual = texto[i]
    siguiente_caracter = texto[i + 1]

    indice_caracter_actual = caracteres_unicos.index(caracter_actual)
    indice_siguiente_caracter = caracteres_unicos.index(siguiente_caracter)

    matriz_transicion[indice_caracter_actual][indice_siguiente_caracter] += 1

  # Comprobar si hay filas con suma cero en la matriz de transición
  filas_con_suma_cero = np.where(matriz_transicion.sum(axis=1) == 0)[0]
  if len(filas_con_suma_cero) > 0:
    # Establecer las filas con suma cero en la matriz de transición a una probabilidad
uniforme
    matriz_transicion[filas_con_suma_cero, :] = 1 / matriz_transicion.shape[1]

  # Normalizar las frecuencias para obtener probabilidades de transición
  matriz_transicion = matriz_transicion / matriz_transicion.sum(axis=1, keepdims=True)

  return matriz_transicion

# Función para calcular la similitud entre dos textos basada en matrices de transición
def calcular_similitud(texto1, texto2):
  # Calcular las matrices de transición para ambos textos
  matriz_transicion1 = calcular_matriz_transicion(texto1)
  matriz_transicion2 = calcular_matriz_transicion(texto2)
```

Daniel Cu Sánchez
A01703613
Actividad final 2. Similitud en textos mediante Cadenas de Markov

```
  # Alinear las matrices de transición al tamaño máximo
  max_size = max(matriz_transicion1.shape[0], matriz_transicion2.shape[0])
  matriz_transicion1 = ajustar_tamaño_matriz(matriz_transicion1, max_size)
  matriz_transicion2 = ajustar_tamaño_matriz(matriz_transicion2, max_size)

  # Calcular la diferencia absoluta entre las matrices de transición
  diferencia_absoluta = np.abs(matriz_transicion1 - matriz_transicion2)

  # Calcular la similitud como la inversa de la suma de las diferencias absolutas
  similitud = 1 / (1 + np.sum(diferencia_absoluta))

  return similitud

# Función para ajustar el tamaño de una matriz de transición
def ajustar_tamaño_matriz(matriz, nuevo_tamaño):
  filas, columnas = matriz.shape
  matriz_ajustada = np.zeros((nuevo_tamaño, nuevo_tamaño))
  matriz_ajustada[:filas, :columnas] = matriz
  return matriz_ajustada

# Función para ingresar la cantidad de textos a comparar
def cantidad_textos(cantidad):
  lista = []
  for x in range(cantidad):
    x = input(f"Ingresa texto [{x + 1}]: ")
    lista.append(x)
  return lista

# Función para calular las similitudes entre los textos ingresados
def calcular_similitudes(lista):
  for x in range(len(lista)):
    similitud = calcular_similitud(lista[0], lista[x])
    print(f"Similitud entre texto[1] y texto[{x + 1}]: {similitud}")


# Función main que da flujo al programa
def main():
  cantidad = int(input("Ingresa cuanto textos quieres comparar: "))
  lista = cantidad_textos(cantidad)
  calcular_similitudes(lista)


# Arranque de programa
if __name__ == "__main__":
  main()
```

Este programa está hecho de forma modularizada.

Utilice la librería numpy como apoyo para el manejo de las matrices.

Lo dividí en 5 funciones de algoritmo y 1 función de arranque.

Daniel Cu Sánchez
A01703613
Actividad final 2. Similitud en textos mediante Cadenas de Markov

calcular_matriz_transicion: Función para calcular la matriz de transición de un texto

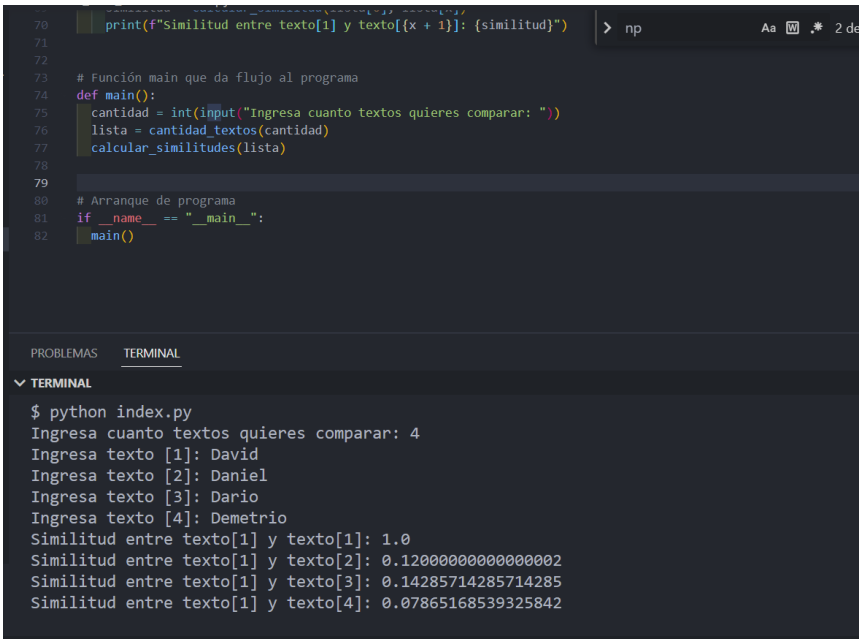
calcular_similitud: Función para calcular la similitud entre dos textos basada en matrices de transición

ajustar_tamaño_matriz: Función para ajustar el tamaño de una matriz de transición

cantidad_textos: Función para ingresar la cantidad de textos a comparar

calcular_similitudes: Función para calular las similitudes entre los textos ingresados

Resultados:



En conclusión, utilizar similitud mediante diagramas de transición ofrece una forma visualmente clara y efectiva de analizar datos, identificar patrones, realizar comparaciones y comunicar información de manera comprensible. Esto puede conducir a una mejor toma de decisiones y una comprensión más profunda de los conceptos o fenómenos estudiados.