Name: Shreyash Rahinj

Batch - A2

Roll no: 371045

PRN: 22110311

**Code :**

import numpy as np

from math import acos,pi,sqrt

from nltk.util import ngrams

# Numerical Data

def euclideanDistance(v1,v2):

dist=0

for i in range(len(v1)):

dist += ((v1[i] - v2[i]) \*\* 2)

return dist \*\* 0.5

def minkowoskiDistance(v1,v2,k):

dist=0

for i in range(len(v1)):

dist += ((v1[i] - v2[i]) \*\* k)

return dist \*\* (1/k)

def manhattanDistance(v1,v2):

dist = 0

for i in range(len(v1)):

dist+= abs(v1[i] - v2[i])

return dist

def supremumDistance(v1,v2):

return np.max(np.abs(np.array(v1) - np.array(v2)))

# Binary Data

def simpleMatchingCoefficient(v1,v2):

M0,M1 = 0,0

for i in range(len(v1)):

if(v1[i]==v2[i]):

M1+=1

else:

M0+=1

return (M1/(M0+M1))

def jacardCoefficient(v1,v2):

v1 = set(v1)

v2 = set(v2)

m = v1.intersection(v2)

n = v1.union(v2)

return len(m)/len(n)

# Textual Data

def jacardSimilarity(s1,s2):

v1 = s1.split()

v2 = s2.split()

v1 = set(v1)

v2 = set(v2)

m = v1.intersection(v2)

n = v1.union(v2)

return len(m)/len(n)

def cosineSimilarity(s1,s2):

s1 = s1.split()

s2 = s2.split()

v1 = set(s1)

v2 = set(s2)

union = v1.union(v2)

v1 = {}

v2 = {}

for i in union:

v1[i] = s1.count(i)

v2[i] = s2.count(i)

num = 0

denom1 = 0

denom2 = 0

for key in v1:

num += v1[key] \* v2[key]

denom1 += (v1[key]) \*\* 2

denom2 += (v2[key]) \*\* 2

return num/(sqrt(denom1) \* sqrt(denom2))

def jaroSimilarity(s1,s2):

m = 0

t = 0

for i in range(len(s1)):

if s1[i] in s2:

m+=1

if s1[i] != s2[i]:

t+=1

return ((m/len(s1)) + (m/len(s2)) + ((m-(t/2))/m))/3

def editDistance(str1, str2, m, n):

if m == 0:

return n

if n == 0:

return m

if str1[m-1] == str2[n-1]:

return editDistance(str1, str2, m-1, n-1)

return 1 + min(editDistance(str1, str2, m, n-1), editDistance(str1, str2, m-1, n), editDistance(str1, str2, m-1, n-1))

def ngramDistance(str1,str2,n):

ngrams1 = set(ngrams(str1.split(),n))

ngrams2 = set(ngrams(str2.split(),n))

intersection = len(ngrams1.intersection(ngrams2))

union = len(ngrams1.union(ngrams2))

return 1 - (intersection/union) if union !=0 else 0

v1 = [69,96]

v2 = [96,69]

print(f" Euclidean Distance : {euclideanDistance(v1,v2)}")

print(f" Manhattan Distance : {manhattanDistance(v1,v2)}")

print(f" Minkowoski Distance : {minkowoskiDistance(v1,v2,3)}")

print(f" Supremum Distance : {supremumDistance(v1,v2)}")

v1 = [1,1,0,0,1,0]

v2 = [1,0,1,1,0,1]

print(f" Simple Matching Coefficient : {simpleMatchingCoefficient(v1,v2)}")

print(f" Jacard Coefficient : {jacardCoefficient(v1,v2)}")

s1 = "Elon Musk"

s2 = "Colon Musk"

print(f" Edit Distance : {editDistance(s1,s2,len(s1.split()),len(s2.split()))}")

print(f" Jacard Similarity : {jacardSimilarity(s1,s2)}")

print(f" Cosine Similarity : {cosineSimilarity(s1,s2)}")

print(f" Jaro Similarity : {jaroSimilarity(s1,s2)}")

print(f" NGram Distance : {ngramDistance(s1,s2,2)}")

**Output :**

A screenshot of a computer

Description automatically generated