**西南大学**

**商贸学院**

**《Python程序设计与算法基础教程》**

**课程作业**

**年级**：2018级

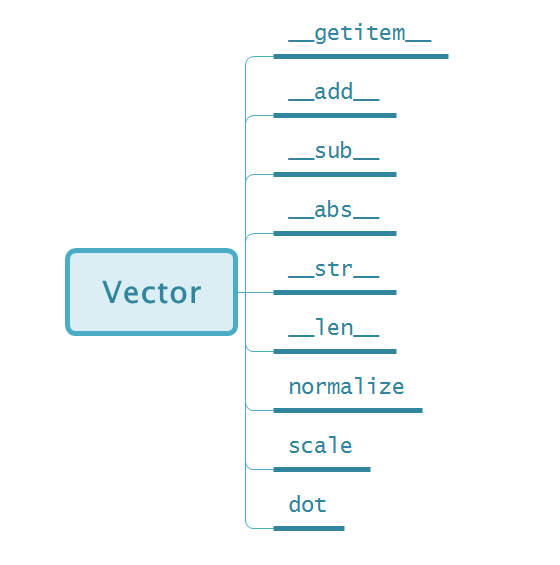
**班级**：计算机科学与技术3班

**学号**：222018603193119

**姓名**：莫健行

# **CS9 文本相似度比较分析**

1. **Vector类的构建与测试**
   1. **Vector类结构：**



* 1. **Vector类源码**

import random, math

class Vector:

def \_\_init\_\_(self, l):

self.\_\_value = l

self.\_\_length = len(l)

def \_\_getitem\_\_(self, i):

return self.\_\_value[i]

def length(self):

return self.length

def scale(self, n):

return Vector([i \* n for i in self.\_\_value])

def \_\_add\_\_(self, b):

result = [self.\_\_value[i] + b[i] for i in range(self.\_\_length)]

return Vector(result)

def \_\_sub\_\_(self, b):

c = [-i for i in b]

return self.\_\_add\_\_(c)

def \_\_abs\_\_(self):

return math.sqrt(self.dot(self))

def dot(self, b):

result = [self.\_\_value[i] \* b[i] for i in range(self.\_\_length)]

return sum(result)

def normalize(self):

return Vector([i / self.\_\_abs\_\_() for i in self.\_\_value])

def \_\_len\_\_(self):

return self.\_\_length

def \_\_str\_\_(self):

return str(self.\_\_value)

* 1. **Vector类测试**

**测试代码：**

if \_\_name\_\_ == "\_\_main\_\_":

v1 = Vector((1,2))

v2 = Vector((2,3))

print("length of v1 is %s" % len(v1))

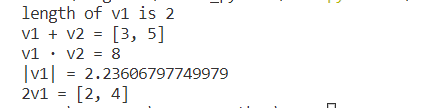
print("v1 + v2 = %s" % (v1+v2))

print("v1 · v2 = %s" % v1.dot(v2))

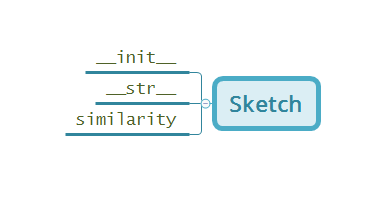
print("|v1| = %s" %abs(v1))

print("2v1 = %s" % v1.scale(2))

**测试结果**

****

1. **Sketch类的设计**
   1. **Sketch类结构：**

****

* 1. **Sketch类源码：**

class Sketch:

def \_\_init\_\_(self, text, k, d):

f = [0 for i in range(d)]

for i in range(len(text) - k):

kg = text[i: i+k]

f[hash(kg) % d] += 1

v = Vector(f)

self.\_\_sketch = v.normalized()

def similarity(self, e):

return self.\_\_sketch.dot(e.\_\_sketch)

def \_\_str\_\_(self):

return str(self.\_\_sketch)

* 1. **Sketch类运行测试：**

**测试代码：**

# test sketch

text1 = "abcdefghijklmn"

text2 = "abcdefghnbklmn"

s1 = Sketch(text1, 2, 4)

s2 = Sketch(text2, 2, 4)

print(s1.similarity(s2))

**运行结果：**

****

1. **数据集随机生成**
   1. **原理：限定数据集，随机生成字符加入字符串中**
   2. **代码：**

def generate():

# add a-z and A-Z

l = [chr(i) for i in range(65, 65 + 26)]

l += [chr(i) for i in range(97, 97 + 26)]

# add space and comma and dot, increase space count to improve proprability

l += [' ', ' ', ' ', ' ', ',', '.']

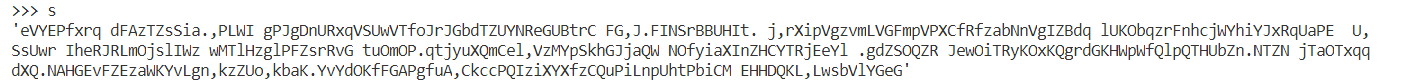
s = ''

for i in range(400):

s += random.choice(l)

return s

* 1. **运行测试：**

****

1. **运行测试脚本：**
   1. **测试流程：**
      1. **生成测试数据集6个，放入data文件夹中**
      2. **生成文本摘要向量**
      3. **相互对比计算相似度，并用表格显示**
   2. **测试数据：**
      1. **测试数据由三组数据组成**
      2. **1.txt与6.txt几乎完全一致**
      3. **3.txt与4.txt有一半的文本不同**
      4. **2.txt与5.txt与其他的文本完全不一致**
   3. **测试代码：**

def compare\_file(path):

# get txt file

l = os.listdir(path)

texts = [t for t in l if t.endswith(".txt")]

#print table head

print(" " \* 15 + "|", end='')

for text in texts:

print(text.center(15) + '|',end="")

print("\n" + "-"\*(len(texts) + 1)\*16)

d = {}

for text in texts:

print(text.center(15),end="|")

for t in texts:

if not d.get(t):

with open(path + '/' + t, 'r') as f:

content = f.read()

d[t] = Sketch(content, 5, 100)

s = d[text]

t = d[t]

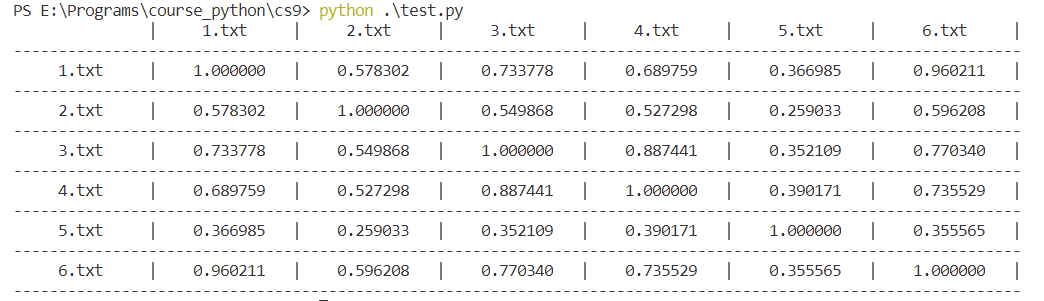
print("{:.6f}".format(s.similarity(t)).center(15), end="|")

print("\n" + "-"\*(len(texts) + 1)\*16)

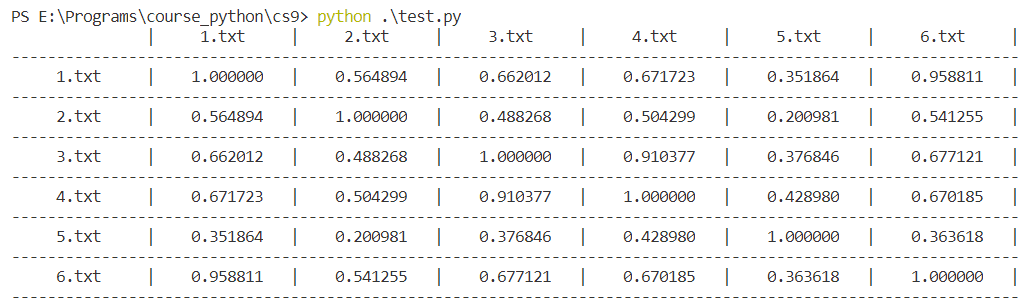
if \_\_name\_\_ == "\_\_main\_\_":

compare\_file("data")

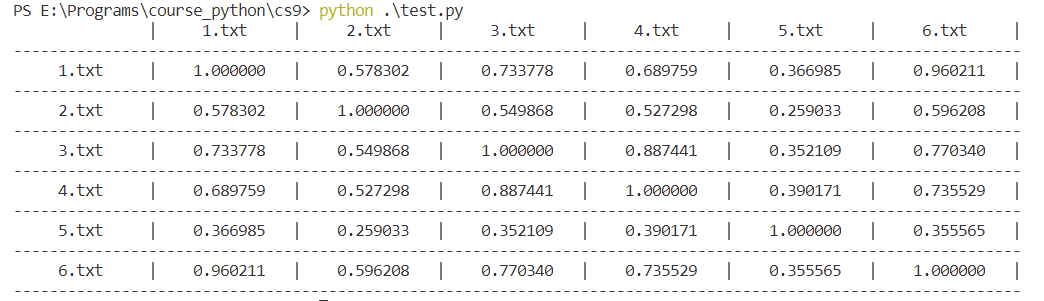
* 1. **运行测试**



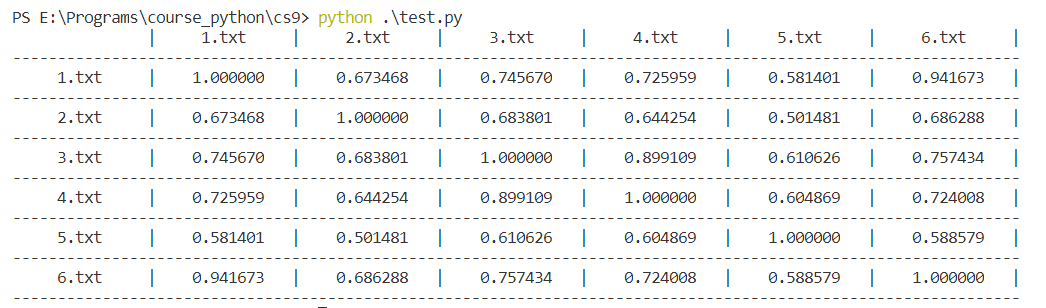
* 1. **运行分析：（k与d的大小选取）**
     1. **对于同一组测试数据，d不变，k的数值变化**
        1. **k = 3**

****

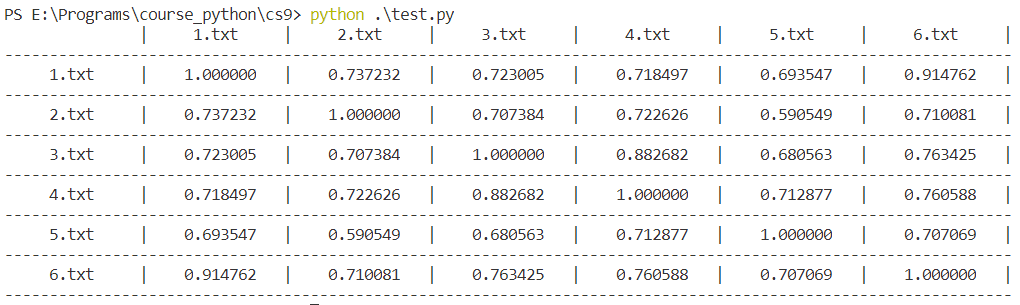
* + - 1. **k = 5**



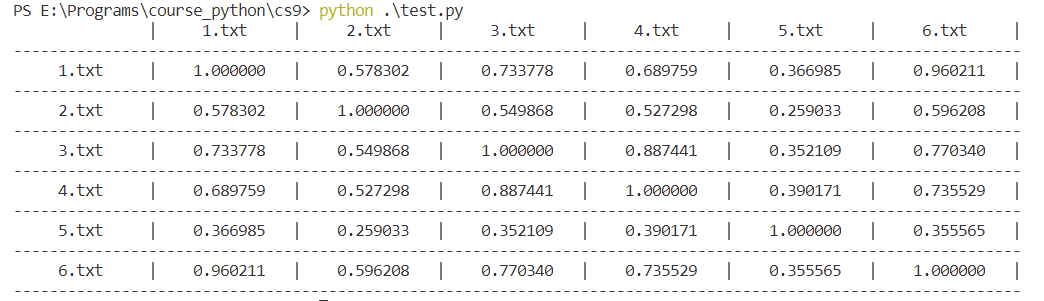
* + - 1. **k = 10**

****

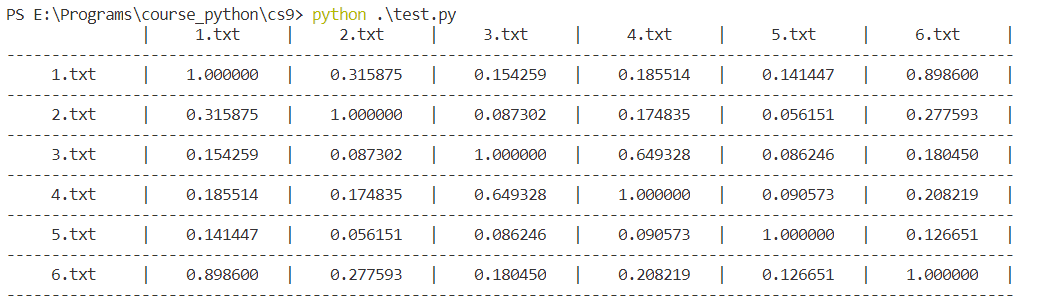
* + - 1. **k = 20**

****

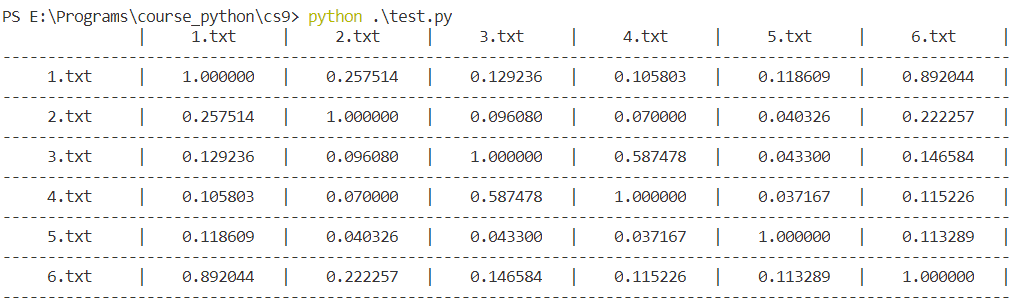
* + 1. **对于同一组数据，k的数值不变，d值变化**
       1. **d = 100**



* + - 1. **d = 1000**

****

* + - 1. **d = 2000**

****

* + 1. **总结：**
       1. **当只有k在变化时，即向量的维度在变化时，k的值越小，其不同文件之间的区分能力越强，具体表现为1.txt与2.txt的相似度随着k值减小而减小，而相似文件1.txt与6.txt的相似度稳定在高水平。**

**原因可能是，随着维度的增加，单个维度的区别对余弦相似度的贡献程度减小，从而导致了区分能力减弱。**

* + - 1. **当只有d在变化时，随着d的增加，区分能力越强。1.txt与2.txt相似度减小，与6.txt相似度稳定于高水平。**

**原因可能是，d作为文本向量的类别，代表着整个文本的特征丰富程度。当d很小时，大部分的文本都会拥有相似的特征，反之文本的不同会更好地展现。**