案例四：使用Python计算余弦相似度

1. 案例目的

• 理解余弦相似度的作用；

• 掌握余弦相似度的原理；

• 掌握余弦相似度的编码实现。

2. 案例内容

案例使用余弦相似度从3篇文本中找出相似度比较高的2篇文本。

余弦相似度是用于衡量文档是否相似的度量。 在数学上，它测量在多维空间中投影的两个矢量的夹角余弦值。余弦相似度是有优势的，因为即使两个相似文档的欧几里德距离很大（文档的大小差异太大），它们仍然可能是相似的。余弦角度越小，余弦相似度越高。

匹配类似文档的常用方法是计算文档之间的公共单词数。但这种方法有一个固有的缺陷，即：随着文档的大小增加，即使文档谈论不同的主题，常用词的数量也趋于增加。余弦相似性有助于克服这一缺陷。

余弦数学公式如下图所示：

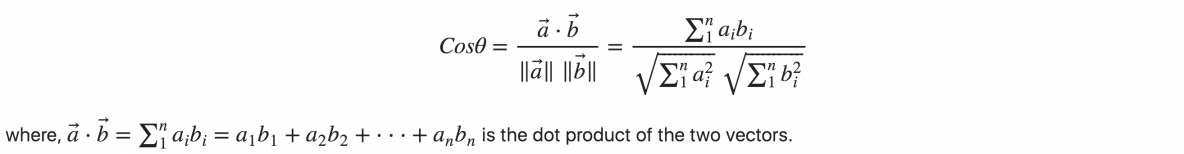


图1 余弦数学公式

3. 案例知识点

• 余弦相似度算法；

• 使用余弦相似度判断文本的相似性。

4. 案例时长

共1学时，具体安排如下：

• 处理数据（0.6学时）

• 计算余弦相似度（0.4学时）

5. 案例实验环境

**•操作系统：**

1）Linux Ubuntu 16.04 64位操作系统

**•软件环境：**

1）Python 3.7

2）sklearn 0.23

**•开发环境与工具：**

1）Spyder 4

6. 案例分析

本案例主要分为以下2部分：

1）处理数据，为计算余弦相似度做准备；

2）计算余弦相似度

7. 案例实验过程

使用Python计算余弦相似度判断文本的相似性，可分为以下4个步骤：

1、导入所需包；

2、导入数据；

3、处理数据；

4、计算余弦相似度。

7.1导入所需包

【代码7-1】cosine\_similarity.py

from sklearn.metrics.pairwise import cosine\_similarity

from sklearn.feature\_extraction.text import CountVectorizer

import pandas as pd

执行代码块。

7.2导入数据

本案例使用3篇文本，所以数据直接以字符串的方式定义即可。

【代码7-2】cosine\_similarity.py

# Define the documents

doc\_trump = "Mr. Trump became president after winning the political election. Though he lost the support of some republican friends, Trump is friends with President Putin"

doc\_trump\_election = "Trump says Russiagate was a witchhunt by political parties. He claimed President Putin is a friend who had nothing to do with the election"

doc\_putin = "Post elections, Vladimir Putin became President of Russia. President Putin had served as the Prime Minister earlier in his political career"

documents = [doc\_trump, doc\_trump\_election, doc\_putin]

执行代码块。

doc\_trump的中文翻译如下：

特朗普在赢得政治选举后成为总统。尽管特朗普失去了一些共和党朋友的支持，但特朗普还是普京总统的朋友。

doc\_trump\_election的中文翻译如下：

特朗普总统说通俄门是来自于别的党派的政治迫害。他声称普京只是个和大选没有关系的朋友。

doc\_putin的中文翻译如下：

选举后，弗拉基米尔普京成为了俄罗斯总统。普京在其政治生涯早期担任总理。

仔细看一下这3篇文本。第一篇文本主要讲的是特朗普成为总统，第二篇文本主要讲的是特朗普通俄门，第三篇文本主要讲的是普京成为总统。这3篇文本出现了好多共同词汇，但是根据我们的理解，第一篇和第二篇更相似一些。

7.3处理数据

计算机是看不懂人类语言的，但是计算机能看懂数字。文本向量化就是将文本用数字化进行表示。文本向量化属于自然语言处理的范畴，这里仅需了解即可。为了简单起见，这里通过统计单词出现的次数来做文本向量化。

【代码7-3-1】cosine\_similarity.py

# Create the Document Term Matrix

count\_vectorizer = CountVectorizer(stop\_words='english')

count\_vectorizer = CountVectorizer()

sparse\_matrix = count\_vectorizer.fit\_transform(documents)

执行代码块。

向量化后是稀疏矩阵，需要转换成DataFrame类型。

【代码7-3-2】cosine\_similarity.py

# OPTIONAL: Convert Sparse Matrix to Pandas Dataframe if you want to see the word frequencies.

doc\_term\_matrix = sparse\_matrix.todense()

df = pd.DataFrame(doc\_term\_matrix,

columns=count\_vectorizer.get\_feature\_names(),

index=['doc\_trump', 'doc\_trump\_election', 'doc\_putin'])

print(df)

执行代码块。

文本向量化后的效果如下图所示：

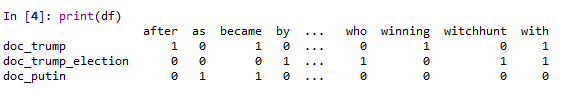


图2 文本向量化后的效果

7.4计算余弦相似度

【代码7-4】cosine\_similarity.py

# Compute Cosine Similarity

print(cosine\_similarity(df, df))

执行代码块。

3篇文本的余弦相似度的结果如图3所示：

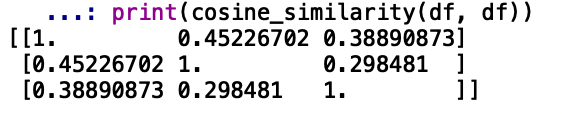


图3 余弦相似度的结果

由上图可见，第1篇与第2篇文本的相似度是0.452，第1篇和第3篇文本的相似度是0.389，第2篇和第3篇文本的相似度是0.298。所以第1篇和第2篇文本更相似一些，和我们的猜想一致。

同学们也可以用其他英文文档来验证余弦相似度的效果。

8. 案例代码

【案例代码】cosine\_similarity.py

# 导入库

from sklearn.metrics.pairwise import cosine\_similarity

from sklearn.feature\_extraction.text import CountVectorizer

import pandas as pd

# Define the documents

doc\_trump = "Mr. Trump became president after winning the political election. Though he lost the support of some republican friends, Trump is friends with President Putin"

doc\_trump\_election = " Trump says Russiagate was a witchhunt by political parties. He claimed President Putin is a friend who had nothing to do with the election"

doc\_putin = "Post elections, Vladimir Putin became President of Russia. President Putin had served as the Prime Minister earlier in his political career"

documents = [doc\_trump, doc\_trump\_election, doc\_putin]

# Create the Document Term Matrix

count\_vectorizer = CountVectorizer(stop\_words='english')

count\_vectorizer = CountVectorizer()

sparse\_matrix = count\_vectorizer.fit\_transform(documents)

# OPTIONAL: Convert Sparse Matrix to Pandas Dataframe if you want to see the word frequencies.

doc\_term\_matrix = sparse\_matrix.todense()

df = pd.DataFrame(doc\_term\_matrix,

columns=count\_vectorizer.get\_feature\_names(),

index=['doc\_trump', 'doc\_trump\_election', 'doc\_putin'])

print(df)

# Compute Cosine Similarity

print(cosine\_similarity(df, df))