Kosinüs benzerliği

Kosinüs benzerliği iki vektör arasındaki benzerliği ölçen matematiksel bir yöntemdir. iki veya daha fazla metinin uzunluklarına bakılmaksızın birbirleri aralarındaki benzerliği bu yöntem sayesinde bulabiliyoruz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

A = [0,1,1,0,1,0,1] 🡪 birinci vektör

B = [0,1,0,0,1,0,1] 🡪 ikinci vektör

**A ve B arasındaki iç çarpımdır ve şu şekilde hesaplanır:**



A . B = (0\*0) + (1\*1)+ (1\*0)+ (0\*0)+ (1\*1)+ (0\*0)+ (1\*1) = 3

**Vektörlerin büyüklüğü şu şekilde hesaplanır:**



||A||= √( (0)^2 + (1)^2 + (1)^2 + (0)^2 + (1)^2 + (0)^2 + (1)^2)) = 2

||B||= √( (0)^2 + (1)^2 + (0)^2 + (0)^2 + (1)^2 + (0)^2 + (1)^2)) = 1.73

Sonuç = = 0,87 bu sonuça göre iki vektör arasındaki benzerlik oranı %87’dir

|  |  |
| --- | --- |
|  | * A ve B vektorlerinin arasındaki Ɵ açısı birbirlerine olan benzerliklerinin oranını ifade eder. * Ɵ açısı büyüdükçe benzerlik oranıda düşmektedir. |

Bu matematiksel yöntemi metinler üzerinde deneyelim.

metin\_a = “en iyi veri bilimciler türkiyede yetişir mi ”

metin\_b = “veri bilimciler her şeye yetişir durumdalar ama yorgunlar ”

Elimizde metin\_a ve metin\_b adında iki tane daha önce atılmış bir twit olsun. İki twit arasındaki benzerliği kosinüs benzerliği yöntemiyle bulalım.

Öncelikle iki twiti birleştirip oluşan yeni twitin benzersiz kelimelerini tabloya yazalım. Daha sonra metin\_a ve metin\_b deki kelimeler toplam twitin içinde varsa “1” yoksa “0” yazalım tabloya

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| toplam | yetişir | şeye | veri | bilimciler | yorgunlar | en | mi | her | iyi | durumdalar | ama | türkiyede |
| Metin1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 1 |
| Metin2 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |

metin\_a = [1,0,1,1,0,1,1,0,1,0,0,1]

metin\_b = [1,1,1,1,1,0,0,1,0,1,1,0]

Artık metinlerimizi vektör olarak ifade edebiliyoruz. Formülde de yerine koyduğumuz taktirde aralarındaki kosinüs benzerliğini bulabileceğiz.

metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

A . B = metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin\_a . metin\_b = 1\*1 + 0\*1 + 1\*1 + 1\*1 + 0\*1 + 1\*0 + 1\*0 + 0\*1 +1\*0 + 0\*1 + 0\*1 + 1\*0 = 3

||A|| = metin içeren bir resim

Açıklama otomatik olarak oluşturuldu

metin\_a = [1,0,1,1,0,1,1,0,1,0,0,1]

||metin\_a|| = = 2.65

metin\_b = [1,1,1,1,1,0,0,1,0,1,1,0]

||metin\_b|| = =2.83

Sonuç = = 0,40 kosinüs benzerlik oranımız

import pandas as pd

import numpy as np

metin1 = "en iyi veri bilimciler türkiyede yetişir mi"

metin2 = "veri bilimciler her şeye yetişir durumdalar ama yorgunlar"

# metinleri boşluklardan ayırıp kelimeleri listeliyoruz

metin1 = metin1.split(" ")

metin2= metin2.split(" ")

# iki listeyi birleştiriyoruz

toplam = metin1 + metin2

# toplam listemizdeki tekrar eden kelimeleri siliyoruz

toplam\_benzersiz = []

for i in toplam:

        if i not in toplam\_benzersiz:

            toplam\_benzersiz.append(i)

# metin  listesindeki kelimeler toplam\_benzersiz listemizde varsa

# 1 yoksa 0 ile değiştirip vektorel hale getiriyoruz

def vektor(vector,toplam\_benzersiz1):

    metin\_v = toplam\_benzersiz1

    for i in toplam\_benzersiz1:

        if i in vector:

            v = toplam\_benzersiz1.index(i)

            metin\_v[v] = 1

        else:

            v = toplam\_benzersiz1.index(i)

            metin\_v[v] = 0

    return metin\_v

metin1\_v = vektor(metin1,toplam\_benzersiz.copy())

metin2\_v = vektor(metin2,toplam\_benzersiz.copy())

# daha net görmek için tablo haline getiriyoruz

# boş bir tablo oluşturuyoruz

df = pd.DataFrame(columns=toplam\_benzersiz, index=["metin11","metin22"])

# vektorlerimizi tabloya ekliyoruz

df.iloc[:1] = [metin1\_v]

df.iloc[1:2] = [metin2\_v]

# iç çarpım

vektor\_ic\_carpim = np.dot(metin1\_v,metin2\_v)

# vektorün büyüklüğünü hesaplıyoruz

def  vektorbuyuklugu(metinv):

    a = 0

    for i in metinv:

        a+=i\*\*2

    return np.sqrt(a)

# bulduğumuz değerleri kosinüs benzerlik formülünde yerine koyuyoruz

kosinus\_benzerlik\_orani = vektor\_ic\_carpim / ( vektorbuyuklugu(metin1\_v)\*vektorbuyuklugu(metin2\_v))

**KISACA TF-IDF NEDİR**

***TERİM SIKLIĞI (TF)***

Bir dökümandaki o kelimenin geçme sayısının o dokümandaki toplam kelime sayısına bölümünden hesaplanan bir değerdir.

**TF(t) = (t teriminin bir belgede görünme sayısı) / (Belgedeki toplam terim sayısı).**

***DOKÜMAN SIKLIĞI (DF)***

Bu sefer doküman sayısının o kelimenin geçtiği doküman sayısına bölümünden hesaplanan bir değerdir

**DF(t) = (Toplam belge sayısı) / (Kelimenin kaç belgede geçtiği).**

***TERS BELGE SIKLIĞI (IDF)***

bir kelimenin ne kadar yaygın kullanıldığını gösteren bir ağırlıktır. Belgeler arasında ne kadar sık ​​kullanılırsa puanı o kadar düşük olur. Log(DF) ile hesaplanır.

IDF değişkeninin kullanılma sebebi dökümanları karakterize etmek. Bir kelimenin belgedeki sıkılığı fazla ise konuyu asıl anlatan veya vurgulayan kelimeleri baskılamaması için bu değişken kullanılır.

**IDF(t)= log( (Toplam belge sayısı) / t kelimesini içeren belge sayısı) )**

**IDF = log( DF )**

import pandas as pd

import difflib

from sklearn.feature\_extraction.text import TfidfVectorizer

from sklearn.metrics.pairwise import cosine\_similarity

df = pd.read\_csv("./data/archive - 2023-02-02T181442.134/imdb\_top\_1000.csv")

# aralarında benzerlik bulabileceğimiz kolonları birleştiriyoruz

df1 = df["Series\_Title"]+" "+df["Genre"]+" "+df["Overview"]+" "+df["Director"]+" "+df["Star1"]+" "+df["Star2"]+" "+df["Star3"]+" "+df["Star4"]

# TF-IDF katsayıları buluyoruz

vectorizer = TfidfVectorizer()

feature\_vectors = vectorizer.fit\_transform(df1)

# birleştirdiğimiz kolonların aralarındaki benzerliği buluyopruz

similarity = cosine\_similarity(feature\_vectors)

movie\_name = "Sherlock"

# verisetimizdeki filmlerin isimlerini listeye çeviriyoruz

list\_of\_all\_titles = df['Series\_Title'].tolist()

# verdiğmiz filmin ismine en yakın filmleri listeliyoruz

find\_close\_match = difflib.get\_close\_matches(movie\_name, list\_of\_all\_titles,n=3)

# yakın isimde bulduğumuz filmeler listesindeki ilk elemanını alıyoruz

close\_match = find\_close\_match[0]

# o filmin verisetimizdeki index'ini buluyoruz

index\_of\_the\_movie = df[df["Series\_Title"] == close\_match].index.values[0]

# o index'i daha önce benzerlik hesabı yapıtığımız listenin içinden buluyoruz

similarity\_score = list(enumerate(similarity[index\_of\_the\_movie]))

# benzerlik listemizdeki o index'e karşılık gelen elemnı alıp sıralıyoruz

sorted\_similar\_movies = sorted(similarity\_score, key = lambda x:x[1], reverse = True)

# sıraladığımız o index değerlerinin ilk elemanı 1 olacaktır çünkü kendisi ile olan benzerlikğini ifade eder

# diğer elemanlar o filme en benzerlik açısından en yakın olan filmleri temsil eder

print('Movies suggested for you : \n')

i = 1

for movie in sorted\_similar\_movies:

  index = movie[0]

  title\_from\_index = df[df.index==index]['Series\_Title'].values[0]

  title\_from\_yonetmen= df[df.index==index]['Director'].values[0]

  title\_from\_star1= df[df.index==index]['Star1'].values[0]

  if (i<19):

    print(i, '.',title\_from\_index, "->",title\_from\_yonetmen,"->", title\_from\_star1)

    i+=1