

دانشکده مهندسی کامپیوتر

دانشگاه اصفهان

**تکلیف چهارم درس بازیابی اطلاعات**

استاد: دکترمحمد مهدی رضاپور

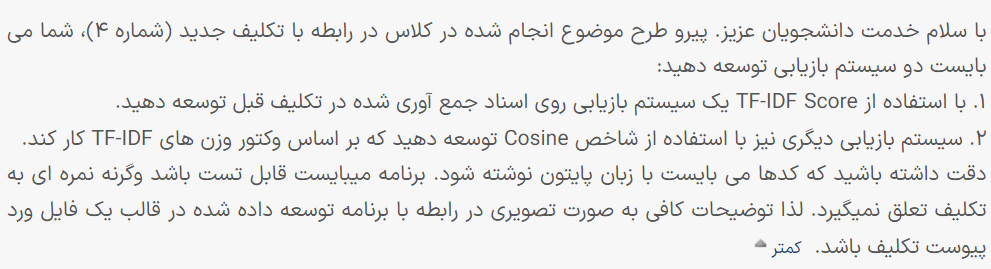
مهروالسادات نوحی

993613061

بهار 1403

شاخص Cosine

صورت تمرین:

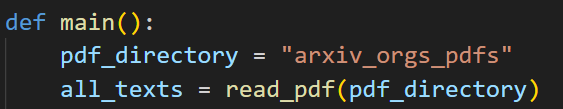


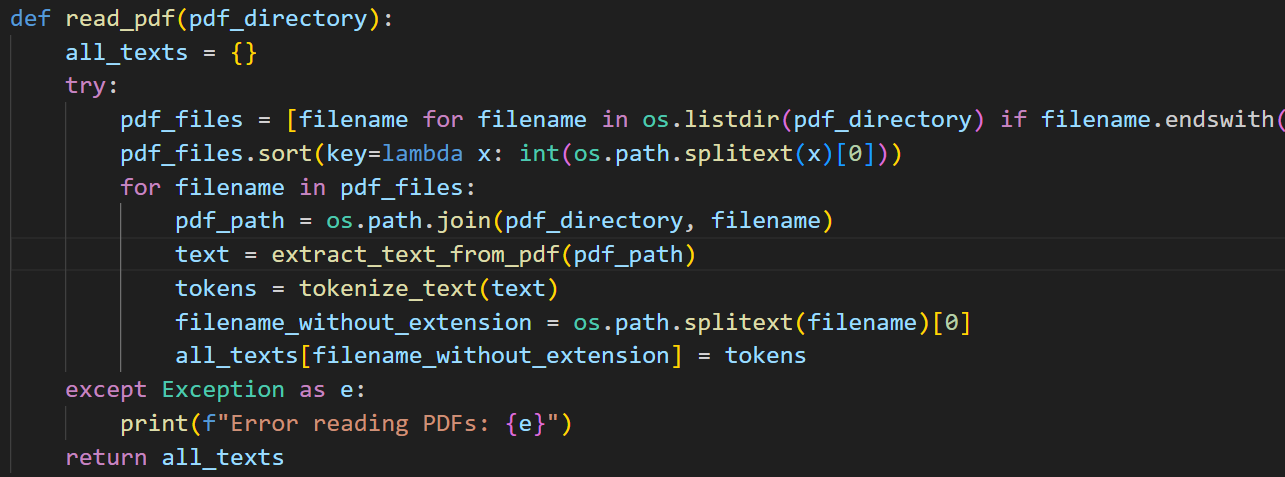
مانند گذشته تمرین را در چند گام تقسیم بندی کرده و جداگانه توضیح خواهم داد. ترجیحا از همان 20 مقاله دانلود شده در تمرین‌‌های قبل برای این تمرین و تمرین بعدی استفاده خواهد شدکه نتایج به صورت واضح‌تر نشان داده شود.

نکته: برخی از گام‌ها کاملا مشابه tf.idf می‌باشد 4 گام اول با بخش قبلی مشترک می‌باشد.

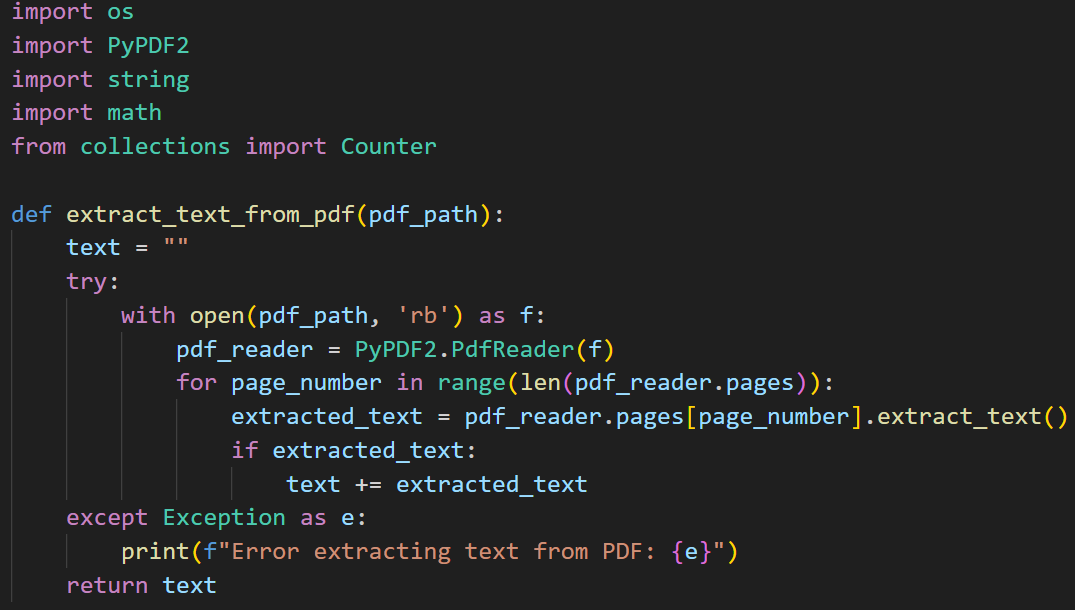
گام اول:

در گام اول از پوشه‌ای که 20 مقاله مورد نظر در آن است باید از هر 20 مقاله شروع به خواندن و استخراج متن کنیم.





این تابع read\_pdf، فایل‌های PDF را از یک دایرکتوری می‌خواند و متن هر فایل را استخراج و توکنیزه می‌کند. سپس نتیجه را به صورت یک دیکشنری ذخیره می‌کند که در آن نام فایل به عنوان کلید و لیست توکن‌ها به عنوان مقدار است.



این تابع extract\_text\_from\_pdf، متن یک فایل PDF را استخراج می‌کند. ابتدا فایل PDF باز می‌شود و سپس متن هر صفحه استخراج و به یک رشته اضافه می‌شود.

با تعریف یک دیکشنری که کلید آن شماره داکیومنت و مقدار آن متن داخل فایل است .

تذکر: من اسم فایل‌ها را برای راحتی کار به مثلا 1.pdf و.. تغییر دادم وبرای استخراج فقط شماره فایل داریم.

در واقع به شکل زیرشده است:

{

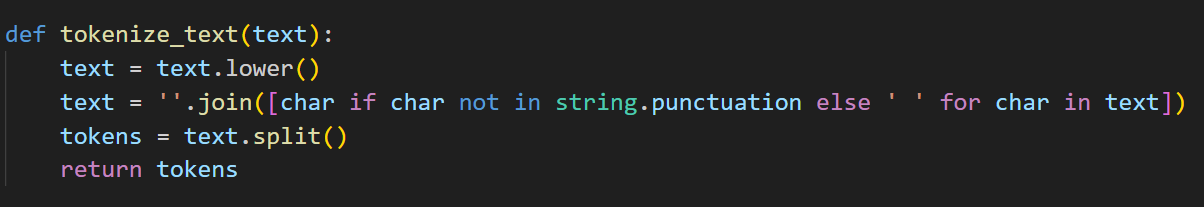
“Filename1”: text1,

“Filename2”:text2,

…{

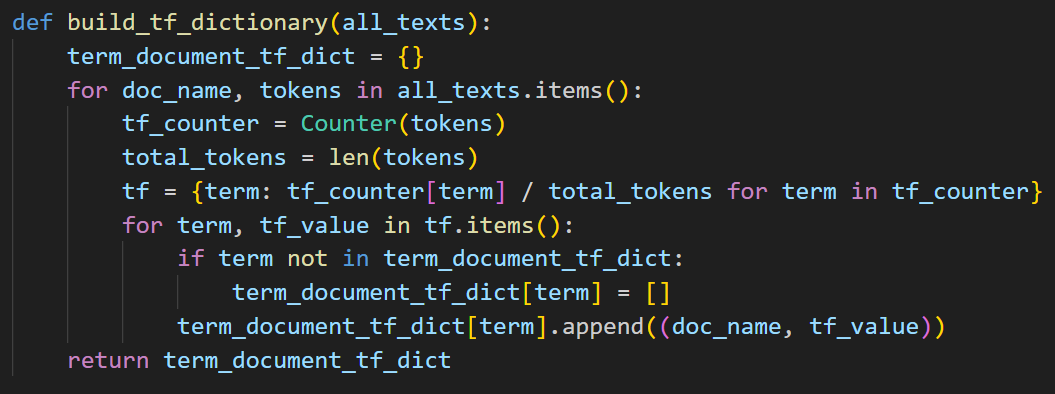
گام دوم:

بعد از اینکه متن‌ها استخراج شده است حال باید توکنایز شود و به اصطلاح هر توکن یک کلمه در نظر بگیریم برای درخواست‌های کاربر.



این تابع tokenize\_text، متن ورودی را به لیستی از توکن‌ها تبدیل می‌کند. ابتدا متن به حروف کوچک تبدیل می‌شود، سپس علائم نگارشی با فاصله جایگزین می‌شوند و در نهایت متن به لیستی از کلمات تقسیم می‌شود.

گام سوم:



این تابع build\_tf\_dictionary، دیکشنری TF را می‌سازد. برای هر داکیومنت، فراوانی نسبی هر توکن (TF) محاسبه می‌شود و در دیکشنری ذخیره می‌شود.به ازای هر داکیومنت، برای همه توکن‌ها در تمام متن استخراج شده یک دیکشنری ساخته و به عنوان مثال کلید آن ترم یا توکن و مقدار آن مقدارtf آن به ازای داکیومنت، و شماره داکیومنت است.

Dict {

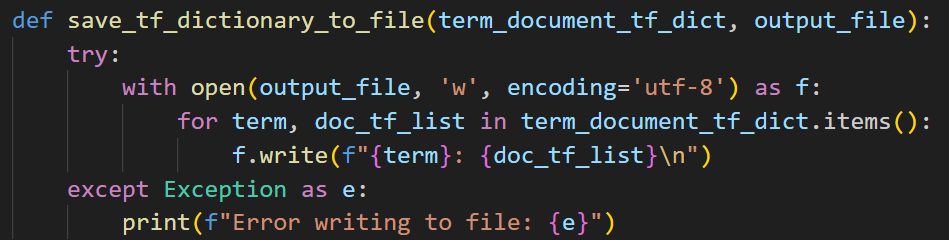
Term1🡪Document\_ID🡪tf #for that document find tf

Term2🡪Document\_ID🡪tf #for that document find tf

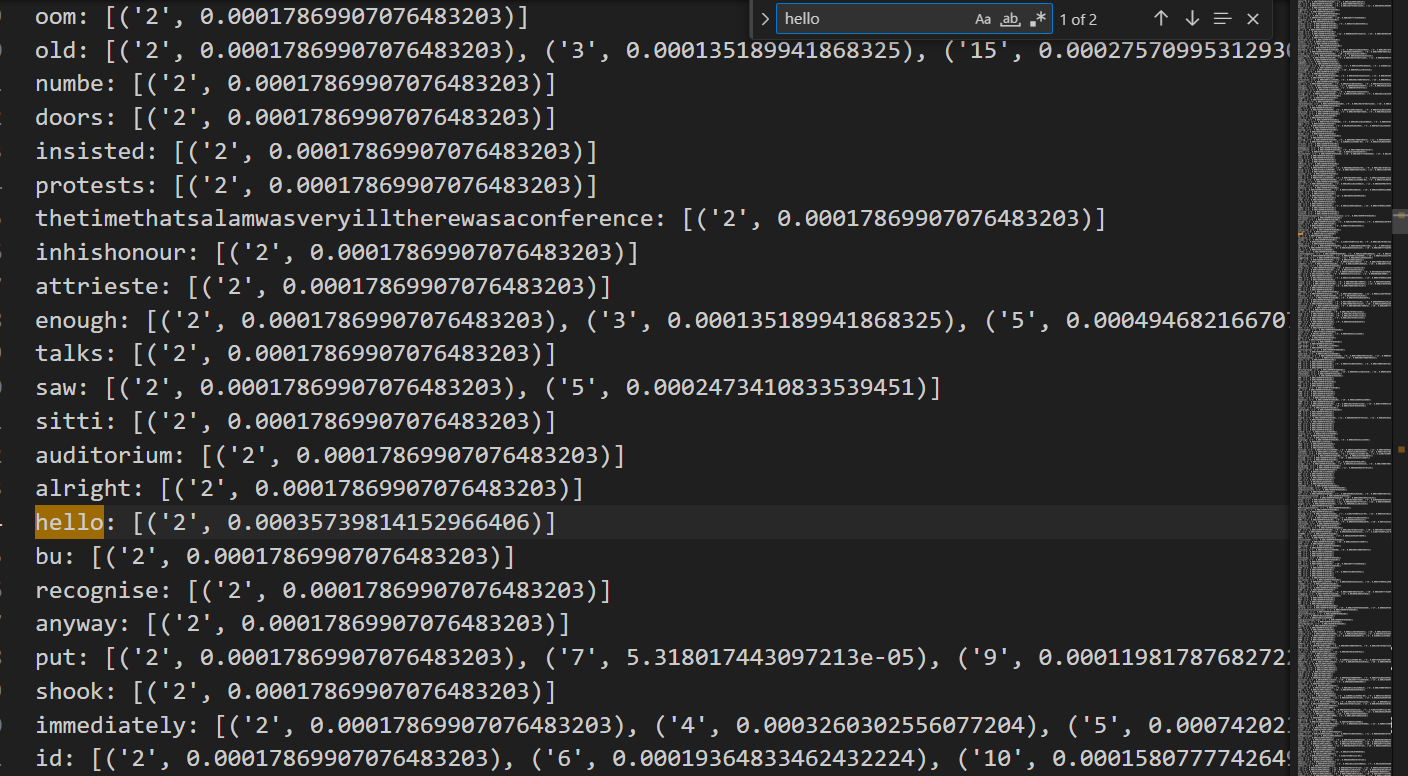
…}

مثال‌ها و توضیحات در بخش قبلی نوشته شده است.

میتوانیم خروجی این گام را در فایل ذخیره کنیم تابع به صورت زیر است:

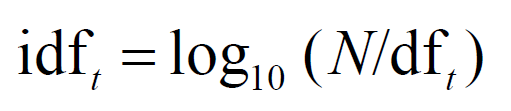


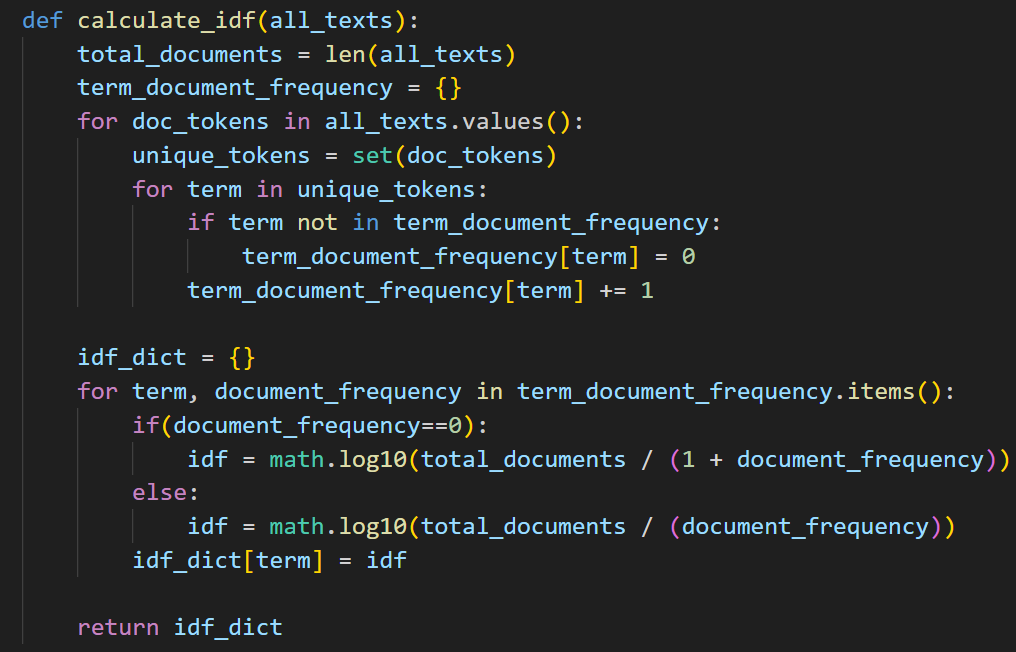
خروجی:



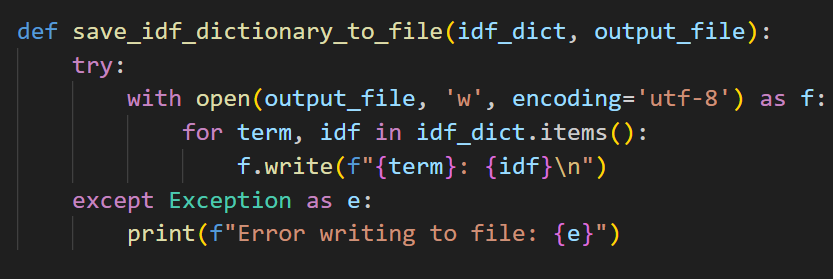
گام چهارم:

در برنامه یک دیکشنری دیگری تعریف کردم که در آن به ازای هر ترم ازهر داکیومنت، idf آن محاسبه شده است که در آینده از آن استفاده کنیم. در واقع این تابع calculate\_idf، مقادیر IDF را برای هر توکن محاسبه می‌کند. فراوانی مستندات برای هر توکن محاسبه می‌شود و سپس IDF محاسبه و در دیکشنری ذخیره می‌شود.

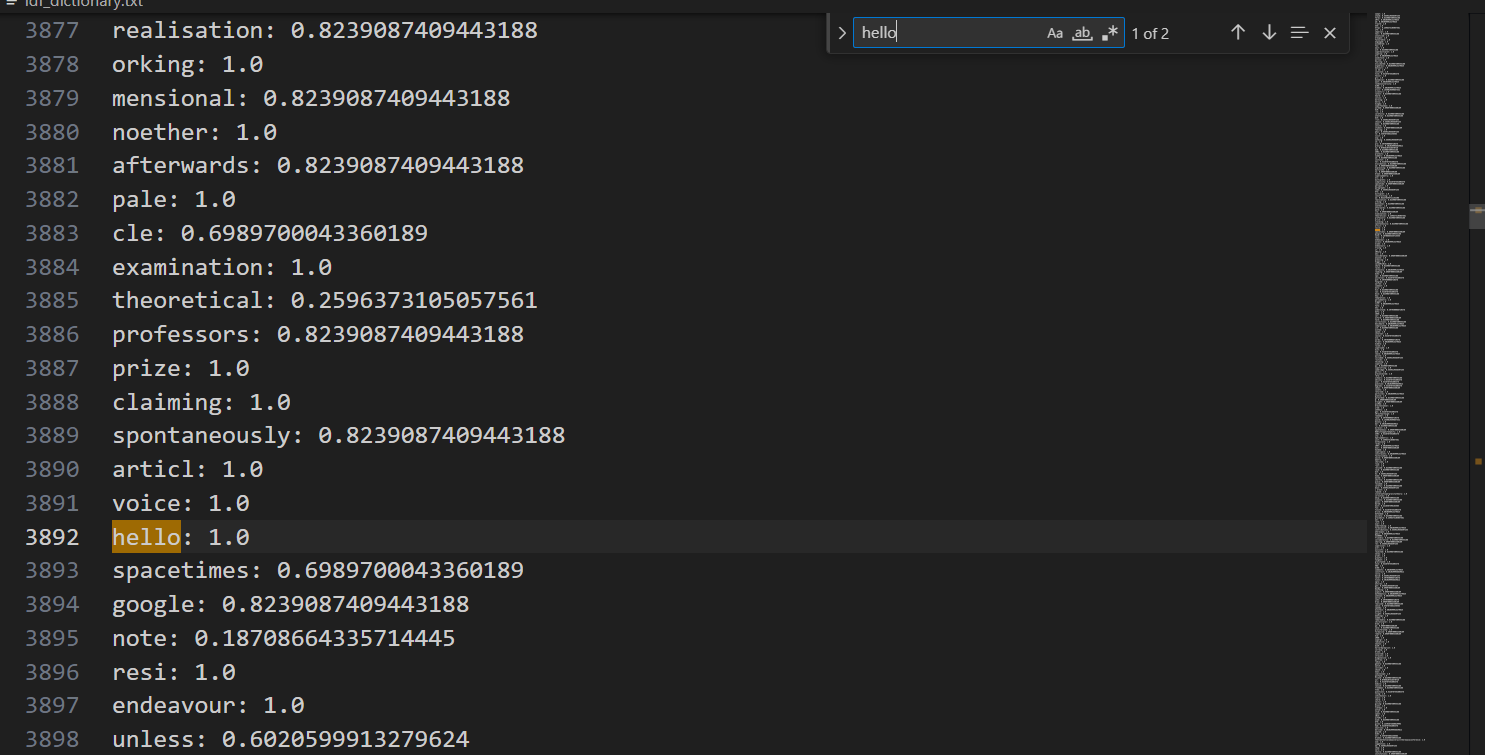




میتوانیم خروجی این گام را در فایل ذخیره کنیم تابع به صورت زیر است:

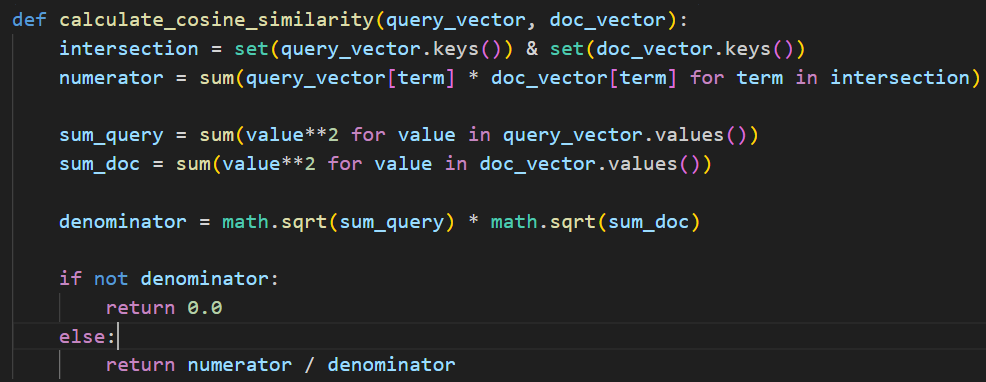


خروجی:



گام پنجم:

از این جا به بعد میریم سراغ محاسبه با شاخص Cosine:



تابع calculate\_cosine\_similarity برای محاسبه شباهت کسینوسی بین دو بردار (وکتور) طراحی شده است. این دو بردار می‌توانند به عنوان نماینده‌ای از دو سند متنی در فضای برداری باشند. شباهت کسینوسی یک مقدار عددی بین -1 و 1 است که نشان‌دهنده میزان مشابهت دو بردار است. هرچه این مقدار به 1 نزدیک‌تر باشد، شباهت بیشتر است و هرچه به -1 نزدیک‌تر باشد، اختلاف بیشتر است.

محاسبه اشتراک کلیدها (واژگان)

  intersection = set(query\_vector.keys()) & set(doc\_vector.keys())

این خط کد، واژگان مشترک بین دو بردار (سند) را پیدا می‌کند. به عبارتی دیگر، کلیدهایی که در هر دو query\_vector و doc\_vector وجود دارند، در مجموعه intersection قرار می‌گیرند.

numerator = sum(query\_vector[term] \* doc\_vector[term] for term in intersection)

صورت کسر با استفاده از جمع حاصل‌ضرب مقادیر واژگان مشترک در هر دو بردار محاسبه می‌شود.

 sum\_query = sum(value\*\*2 for value in query\_vector.values())

 sum\_doc = sum(value\*\*2 for value in doc\_vector.values())

در این دو خط کد، مجموع مربعات مقادیر هر بردار محاسبه می‌شود. این مقادیر برای محاسبه قدر مطلق هر بردار نیاز است.

denominator = math.sqrt(sum\_query) \* math.sqrt(sum\_doc)

مخرج کسر با استفاده از ضرب قدر مطلق (Norm) هر دو بردار محاسبه می‌شود. قدر مطلق هر بردار با جذر مجموع مربعات مقادیر آن بردار بدست می‌آید.

  if not denominator:

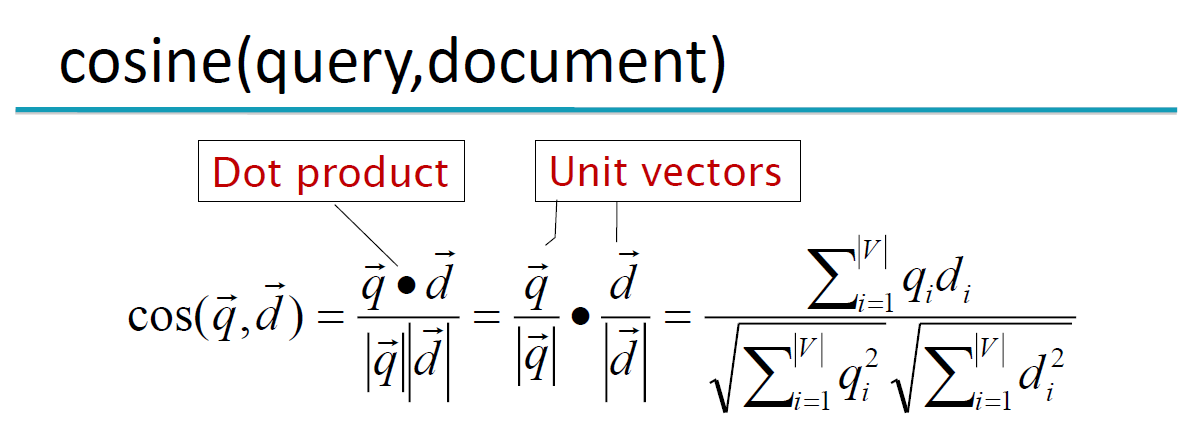
        return 0.0

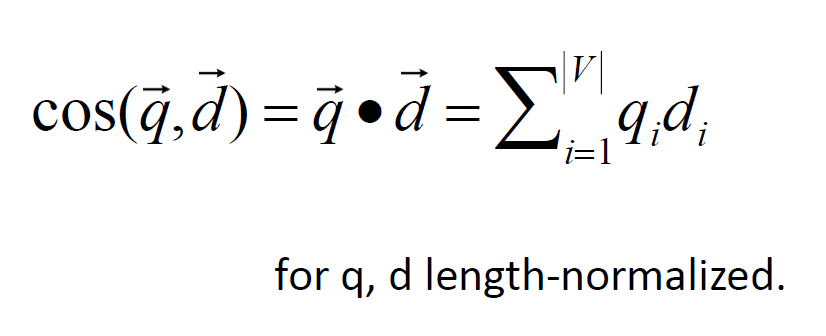
    else:

        return numerator / denominator

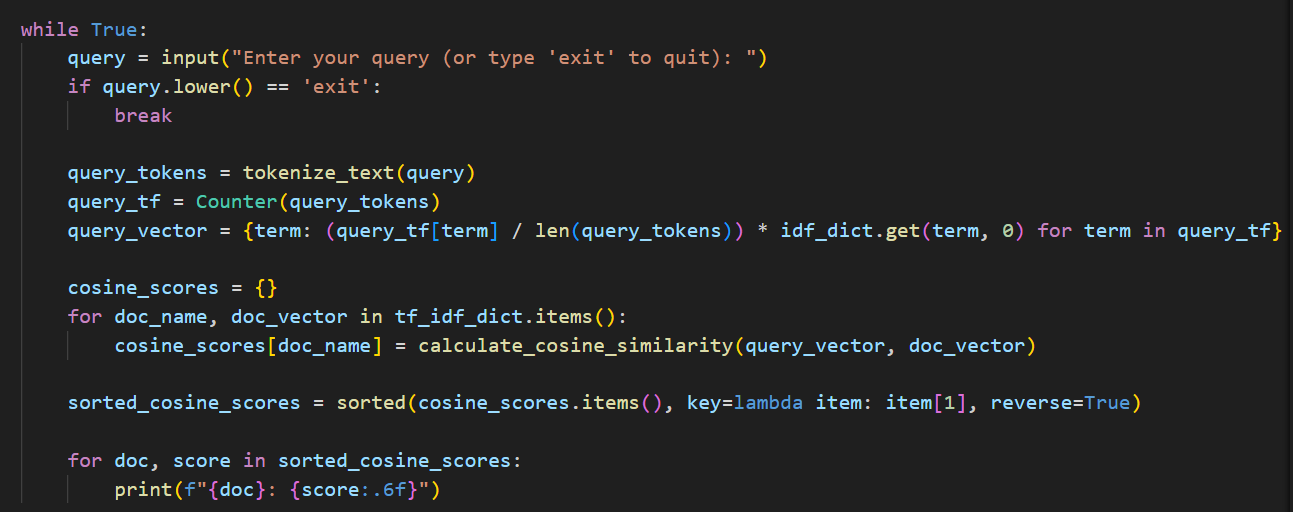
در این بخش، ابتدا بررسی می‌شود که مخرج کسر صفر نباشد. اگر مخرج صفر باشد، شباهت کسینوسی برابر با صفر است. در غیر این صورت، صورت کسر تقسیم بر مخرج شده و شباهت کسینوسی محاسبه می‌شود.

به طور خلاصه، این تابع ابتدا واژگان مشترک بین دو بردار را پیدا کرده، سپس حاصل‌ضرب مقادیر این واژگان را جمع می‌کند (صورت کسر)، و در نهایت مجموع مربعات مقادیر هر بردار را محاسبه کرده و قدر مطلق هر بردار را بدست می‌آورد (مخرج کسر). با تقسیم صورت بر مخرج، شباهت کسینوسی بدست می‌آید.



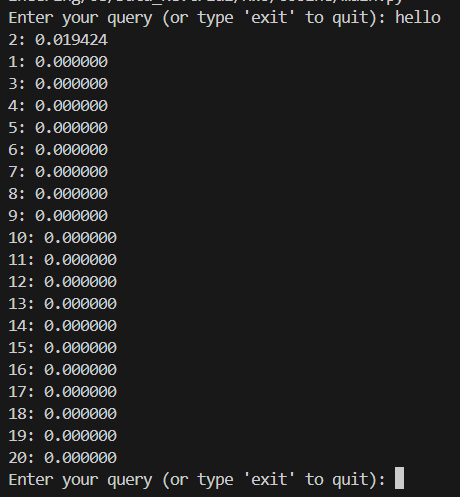


و در انتها داکیومنت‌هایی با بیش‌ترین ضریب برای برگرداننده می‌شود.



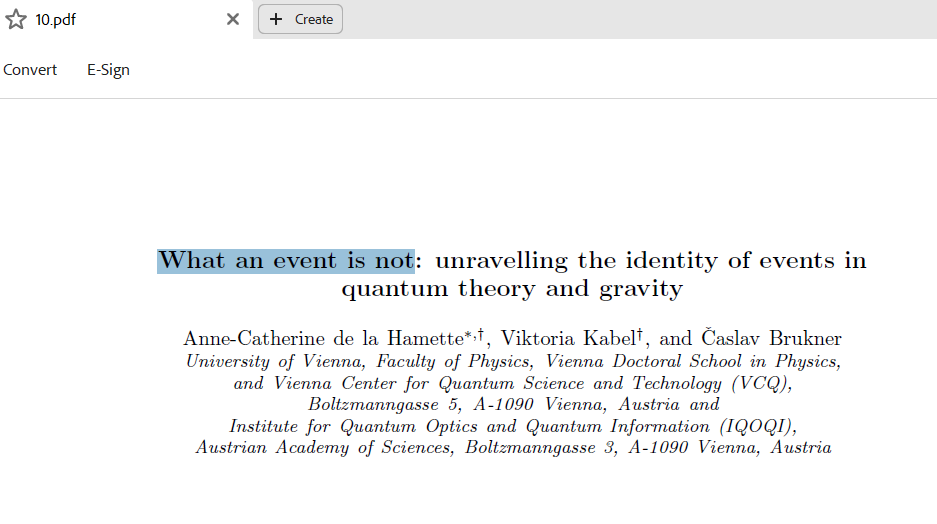
گام ششم:

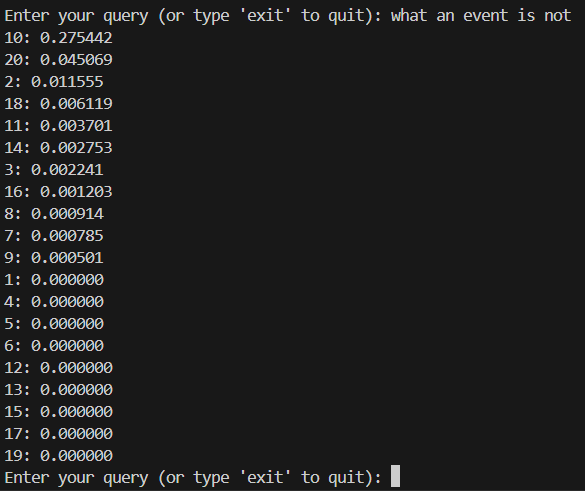
در این گام با نوشتن چند تست برنامه را تست می‌کنیم:



در بخش قبل نیز داکیومنت 2 برگردانده شد.

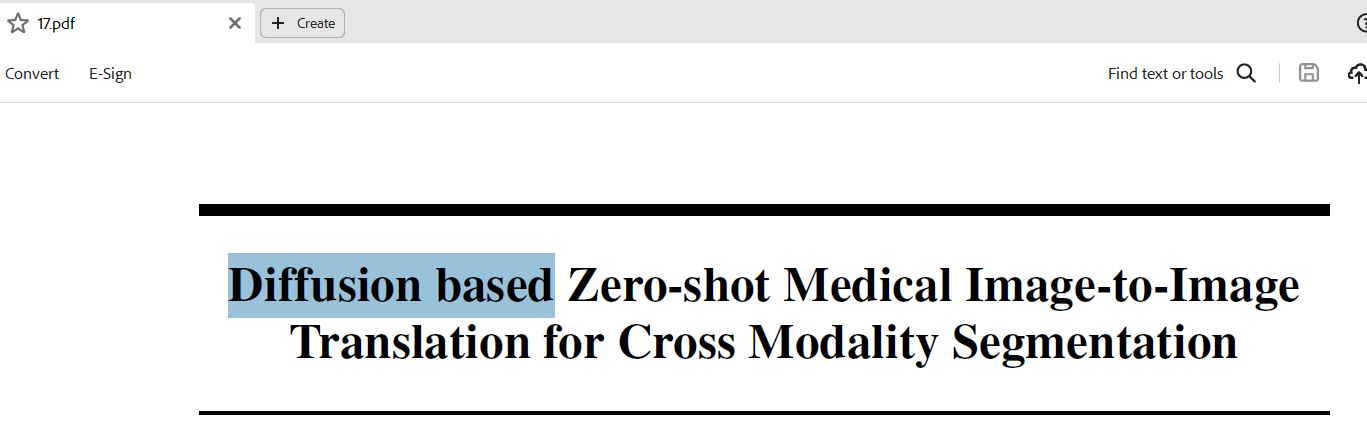
مثال بعدی:

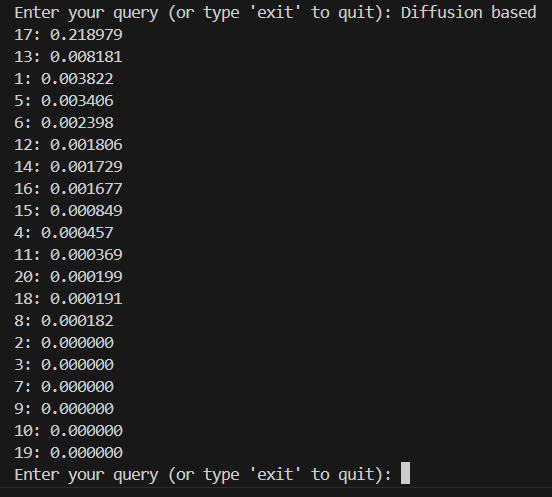




داکیومنت 10 به عنوان اولین داکیونت آورده شده است.

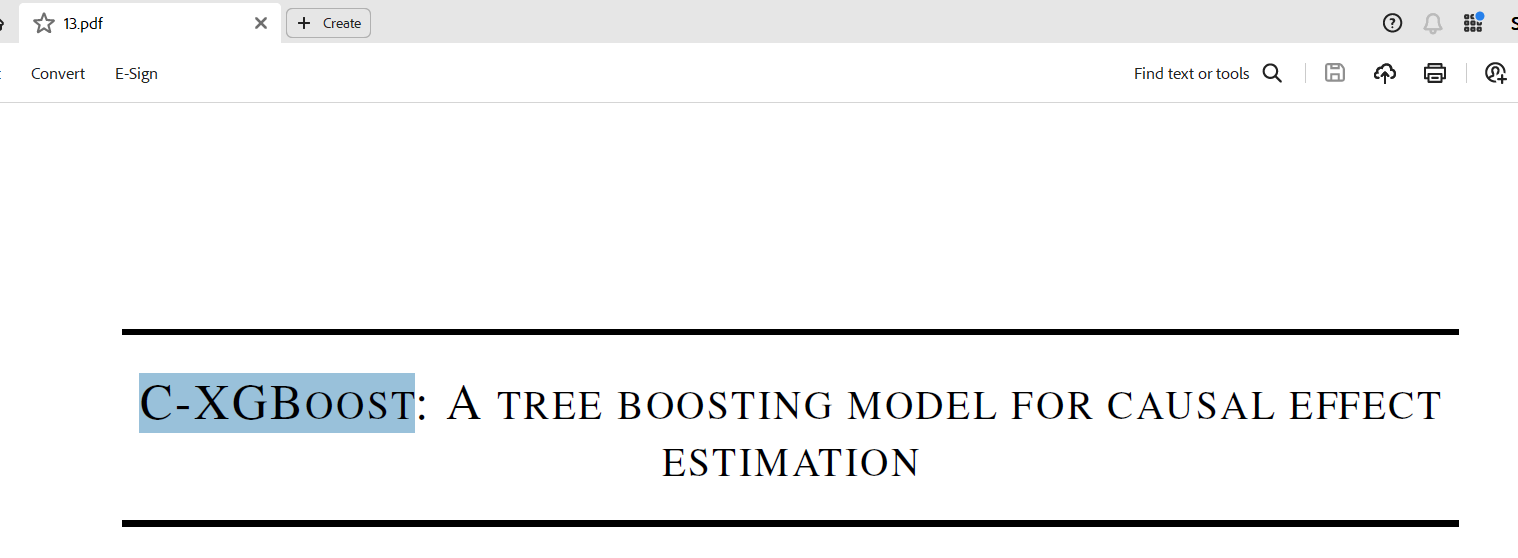
مثال بعدی:

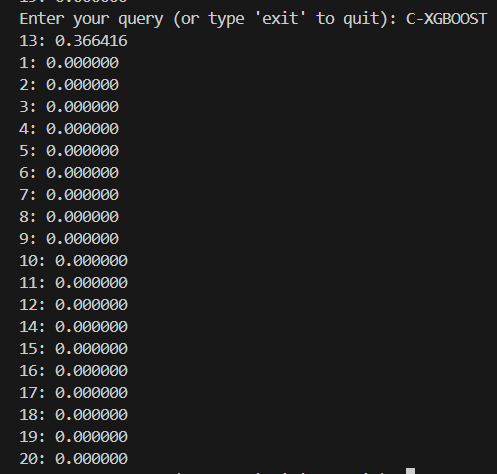




داکیومنت 17 در اولین داکیومنت است.

مثال بعدی:





چون کلمه یکم عجیب و ی جورایی یونیک بوده فقط داکیومنت 13 بازگرداننده شده است. تست‌های دیگر نیز درست بازگرداننده شده است.