

دانشکده مهندسی کامپیوتر

دانشگاه اصفهان

**تکلیف چهارم درس بازیابی اطلاعات**

استاد: دکترمحمد مهدی رضاپور

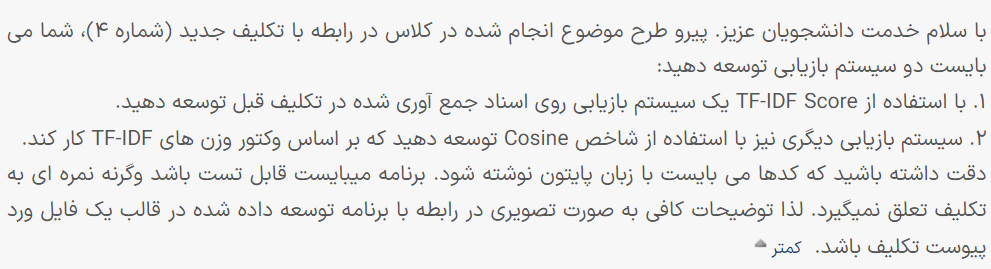
مهروالسادات نوحی

993613061

بهار 1403

TF-IDF Score

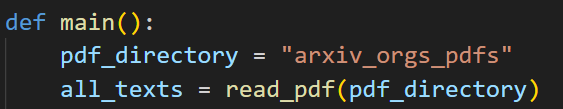
صورت تمرین:

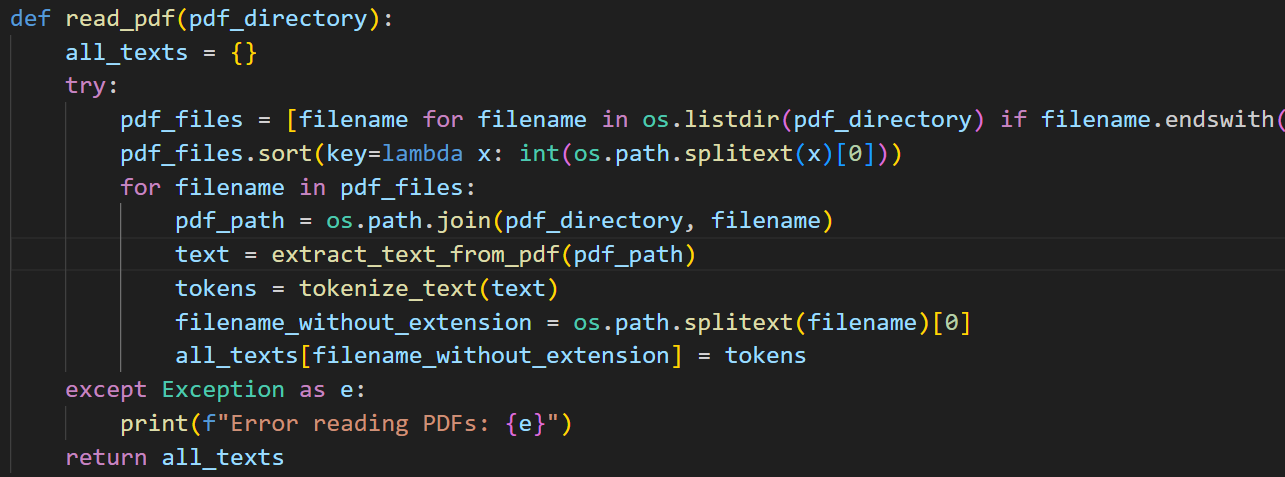


مانند گذشته تمرین را در چند گام تقسیم بندی کرده و جداگانه توضیح خواهم داد. ترجیحا از همان 20 مقاله دانلود شده در تمرین‌‌های قبل برای این تمرین و تمرین بعدی استفاده خواهد شدکه نتایج به صورت واضح‌تر نشان داده شود.

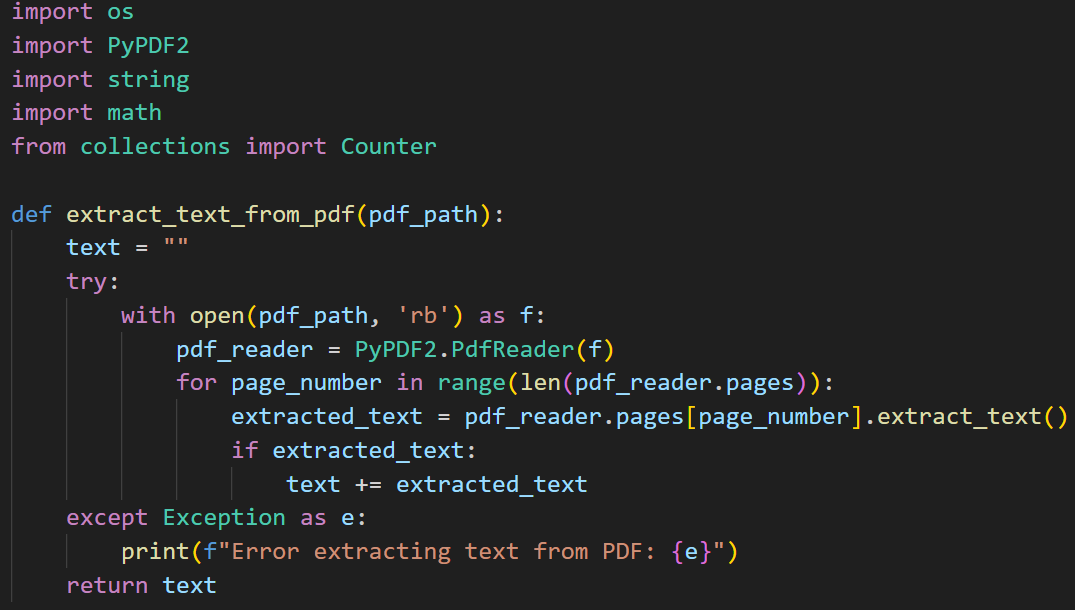
گام اول:

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:

{

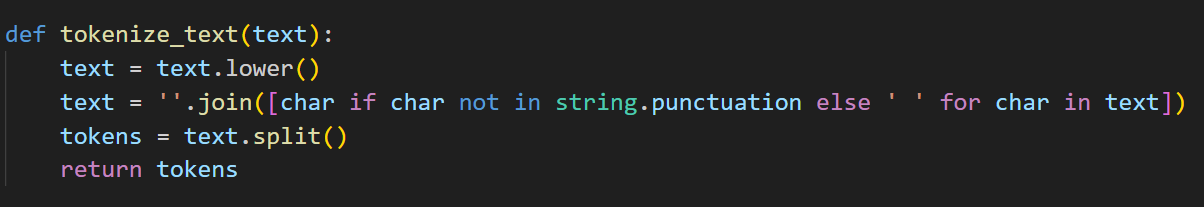
“Filename1”: text1,

“Filename2”:text2,

…{

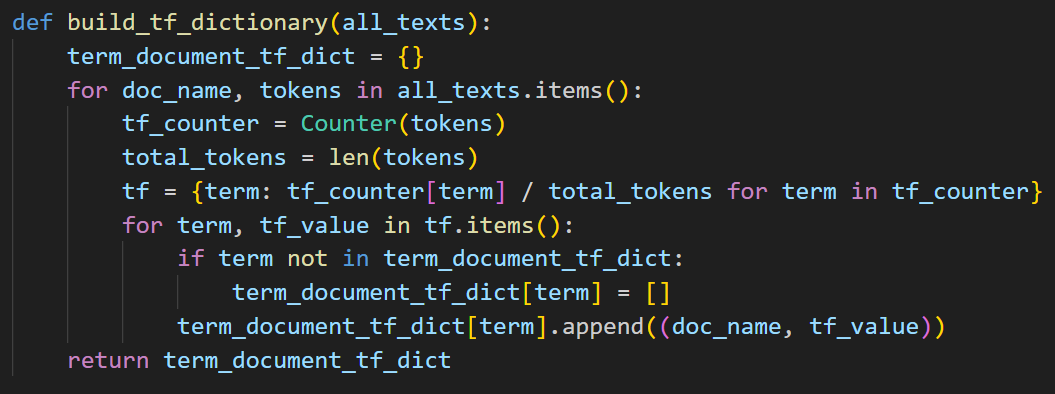
گام دوم:

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

…}

بریم یک مثال ببینیم:

**doc1**: "the quick brown fox"

**doc2**: "the quick brown"

**doc3**: "the quick

این همان all\_texts است که بعنوان ورودی تابع داده می‌شود.

all\_texts = {"doc1": ["the", "quick", "brown", "fox"], "doc2": ["the", "quick", "brown"], "doc3": ["the", "quick"] }

در اینجا دیکشنری رامی‌سازیم:

term\_document\_tf\_dict = {}

برای doc1:

doc\_name = "doc1"

tokens = ["the", "quick", "brown", "fox"]

tf\_counter = Counter(tokens) # {'the': 1, 'quick': 1, 'brown': 1, 'fox': 1}

total\_tokens = 4

tf = {term: tf\_counter[term] / total\_tokens for term in tf\_counter}

# {'the': 0.25, 'quick': 0.25, 'brown': 0.25, 'fox': 0.25}

به دیکشنری اضافه می‌کنیم:

term\_document\_tf\_dict = { 'the': [('doc1', 0.25)], 'quick': [('doc1', 0.25)], 'brown': [('doc1', 0.25)], 'fox': [('doc1', 0.25)] }

برای doc2:

doc\_name = "doc2"

tokens = ["the", "quick", "brown"]

tf\_counter = Counter(tokens) # {'the': 1, 'quick': 1, 'brown': 1}

total\_tokens = 3

tf = {term: tf\_counter[term] / total\_tokens for term in tf\_counter}

# {'the': 0.3333, 'quick': 0.3333, 'brown': 0.3333}

دیکشنری آپدیت می‌شود:

term\_document\_tf\_dict = {

'the': [('doc1', 0.25), ('doc2', 0.3333)],

'quick': [('doc1', 0.25), ('doc2', 0.3333)],

'brown': [('doc1', 0.25), ('doc2', 0.3333)],

'fox': [('doc1', 0.25)]

}

برای doc3:

doc\_name = "doc3"

tokens = ["the", "quick"]

tf\_counter = Counter(tokens) # {'the': 1, 'quick': 1}

total\_tokens = 2

tf = {term: tf\_counter[term] / total\_tokens for term in tf\_counter}

# {'the': 0.5, 'quick': 0.5}

مجدد دیکشنری آپدیت می‌شود. و این نتیجه نهایی است چون پردازش در این مثال تکمیل شد.

**term\_document\_tf\_dict = {**

**'the': [('doc1', 0.25), ('doc2', 0.3333), ('doc3', 0.5)],**

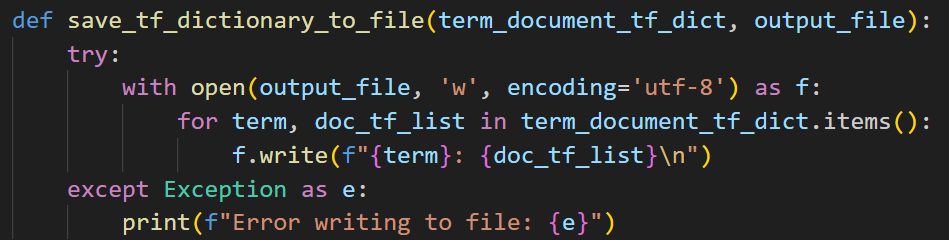
**'quick': [('doc1', 0.25), ('doc2', 0.3333), ('doc3', 0.5)],**

**'brown': [('doc1', 0.25), ('doc2', 0.3333)],**

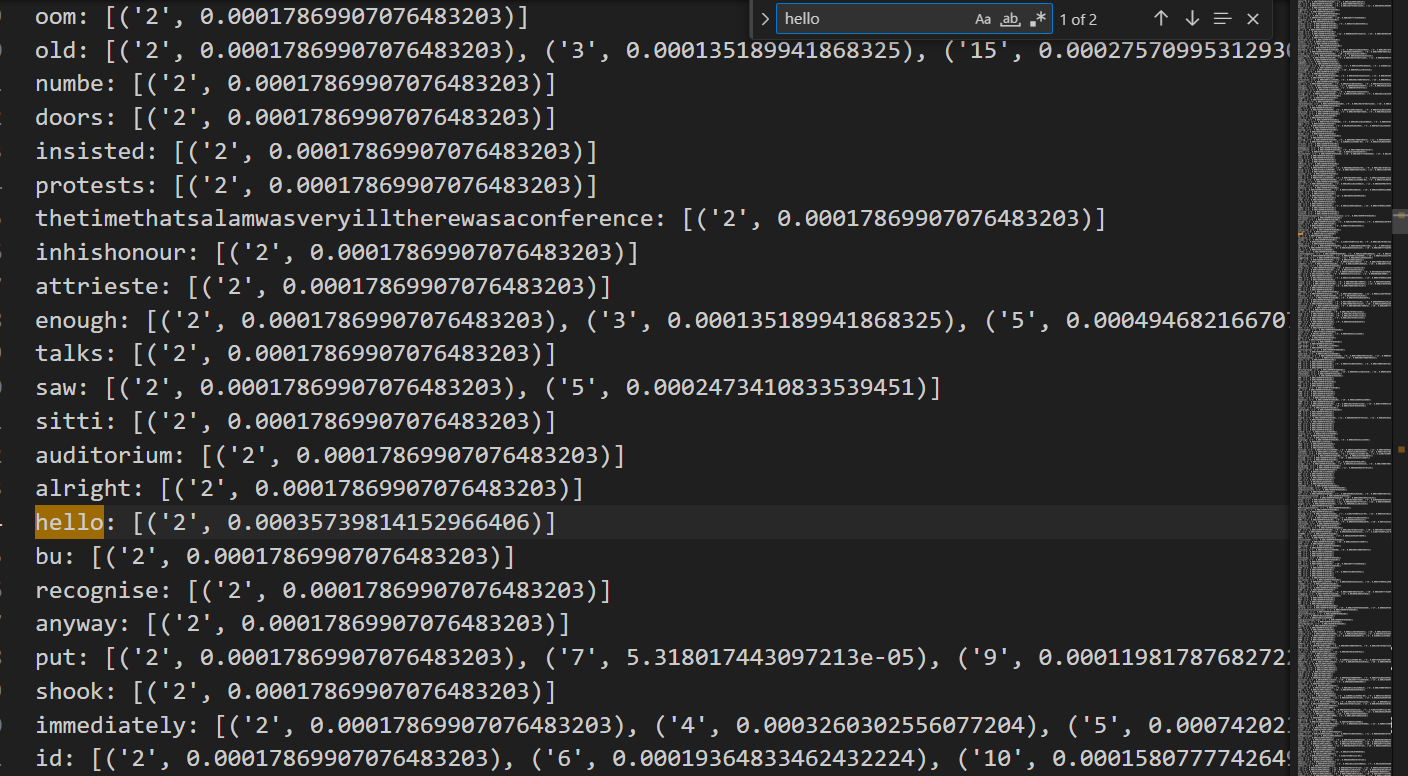
**'fox': [('doc1', 0.25)]**

**}**

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

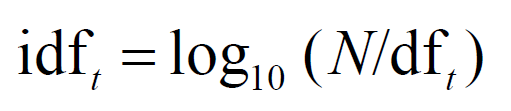


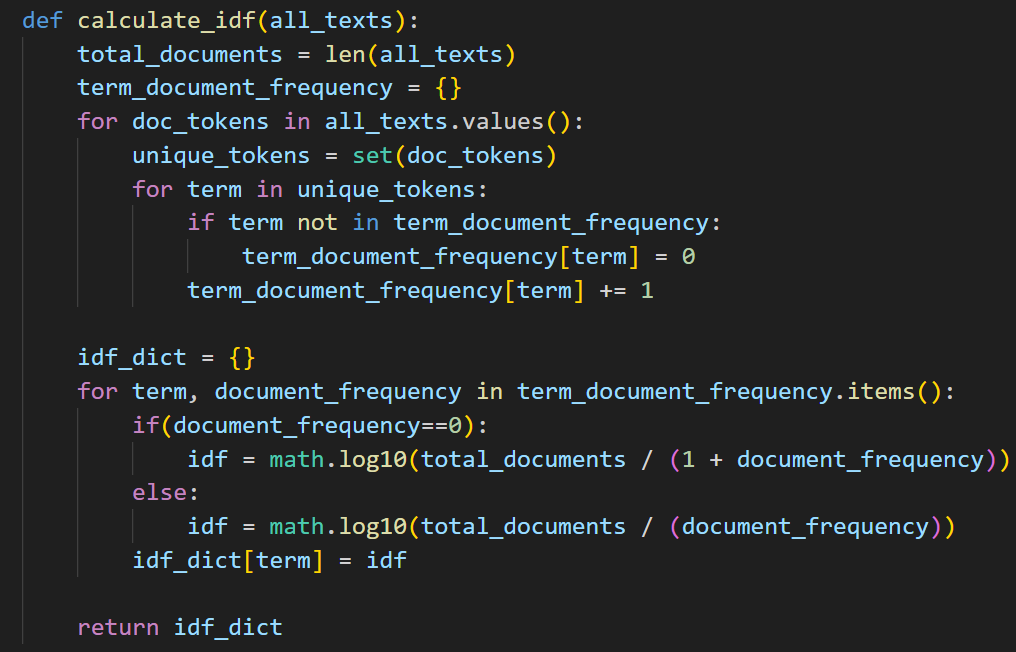
خروجی:



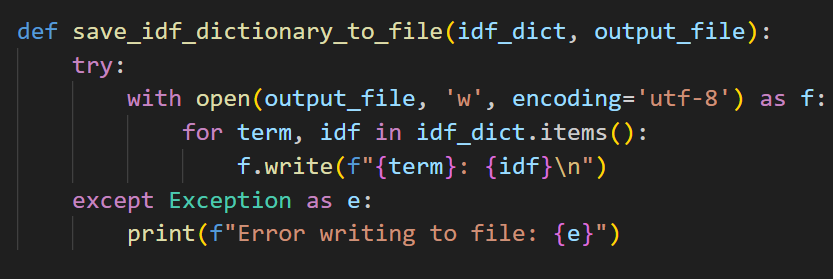
گام چهارم:

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.

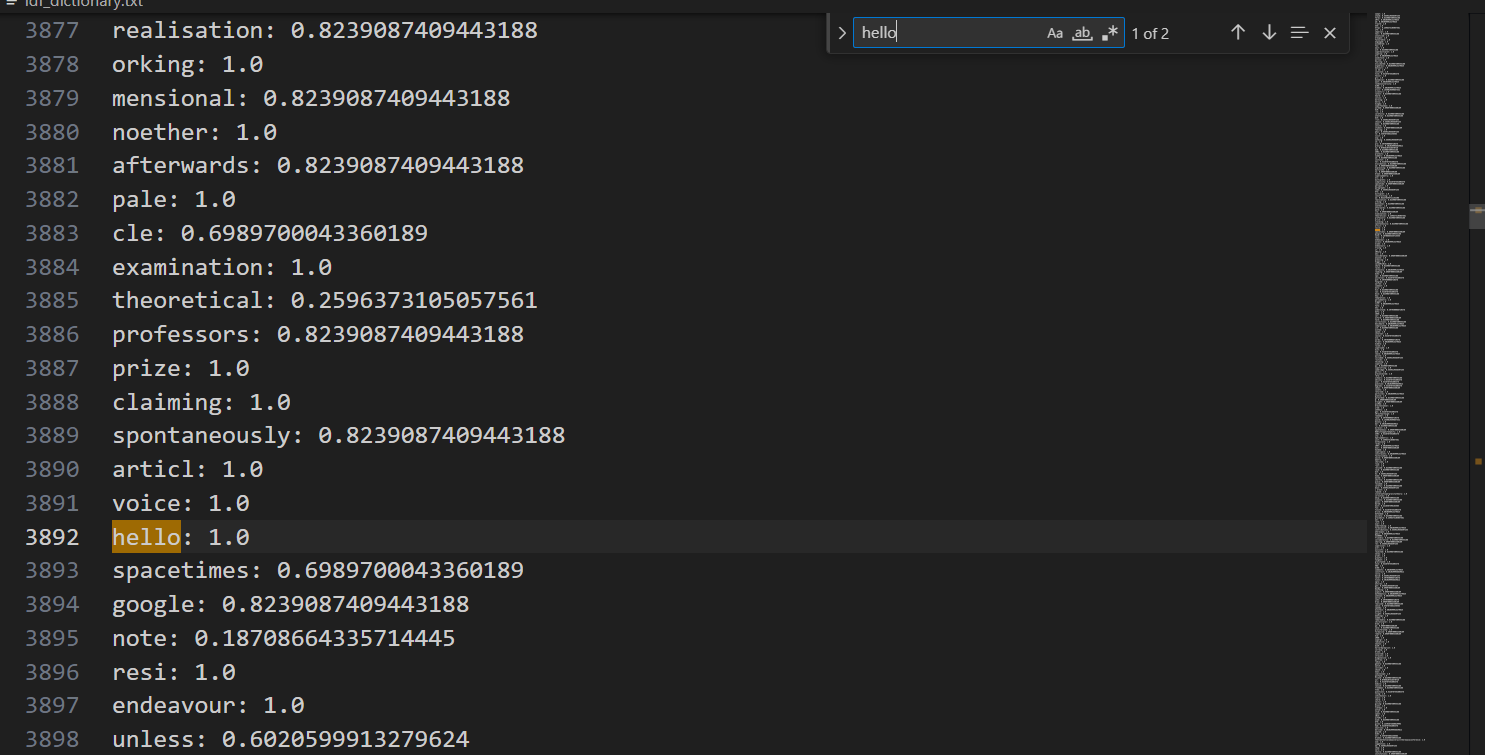




میتوانیم خروجی این گام را در فایل ذخیره کنیم تابع به صورت زیر است:

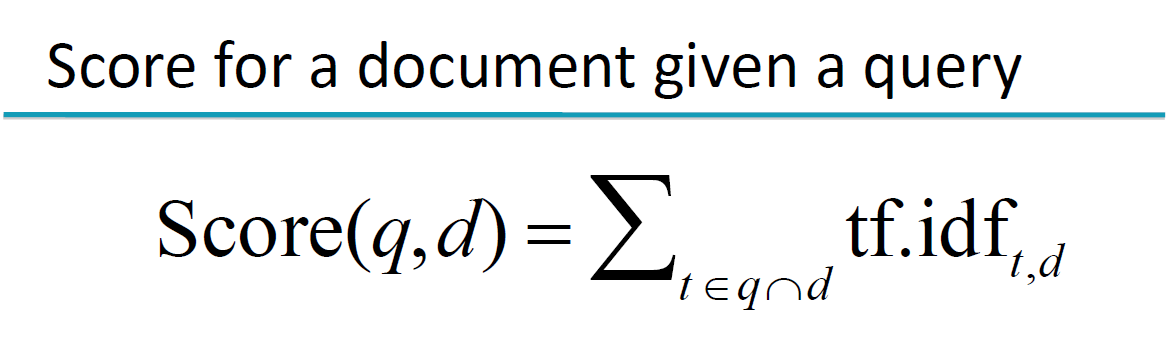


خروجی:

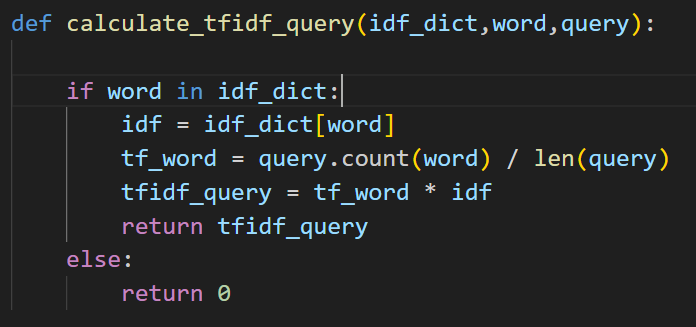


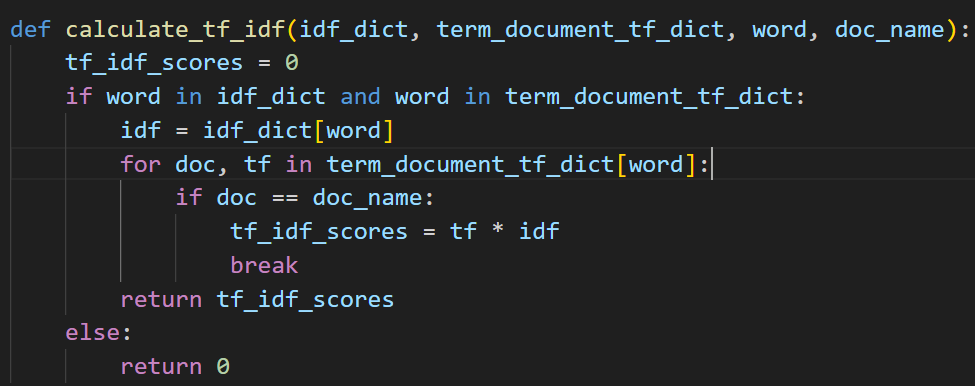
گام پنجم:

دیکشنری‌ها آماده است و بریم سراغ کوعری. طبق فرمول اسلاید باید tf.idf برای هم کوعری و هم برای کل هر ترم کوعری محاسبه شده ضرب شده این مقدار و با هم جمع شوند.



پس از این رو دو تابع تعریف می‌کنیم یکی برای محاسبه tf.idf خوده کوعری و یکی tf.idf کلی که حاصل ضرب این دو برای هر 20 مقاله محاسبه شده و بالاترین مقدار از لحاظ رنک نشان داده می‌شود.





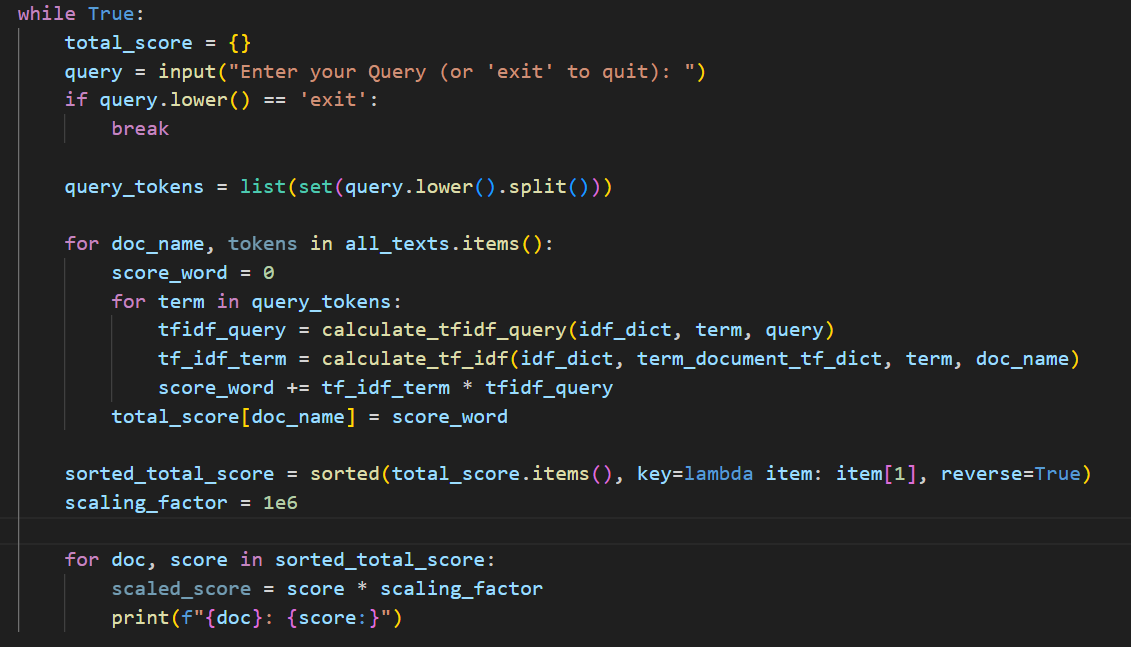
کاری که من کردم چون بردارها را داریم اومدم و برای هر 20 تا مقاله دونه دونه یک حلقه بیرونی واسه مقاله و یک حلقه درونی به ازای هر توکن در کوعری این مقدار محاسبه و با تعریف یک دیکشنری جدید به نام total\_score با کلید داکیومنت، و مقدار tf.idf کل واسه اون مقاله به ازای کوعری مقدار را قرار دادم.

مثال:

Query: “hello world”

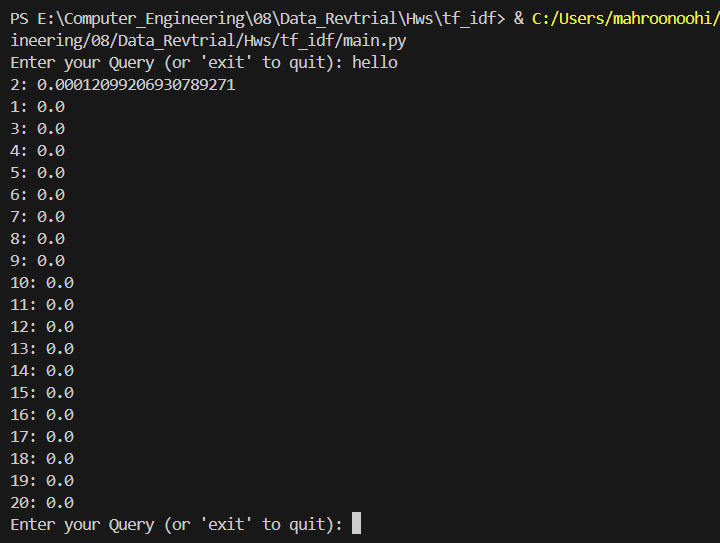
Document1: (tf.idf(q)\*tf.idf)(hello)+ (tf.idf(q)\*tf.idf)(world)

……#for all 20 documents



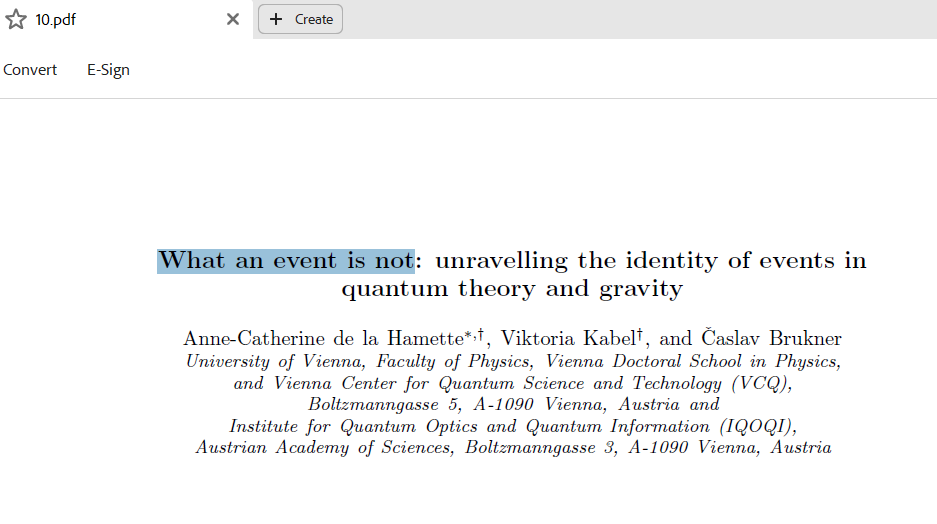
گام ششم:

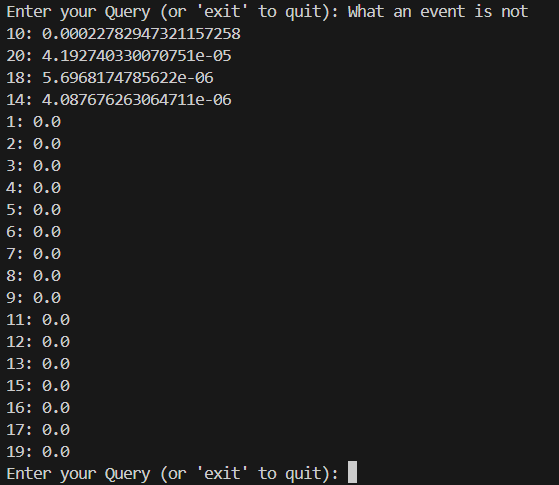
در این گام میریم سراغ تست برنامه :



رفتم چک کردم و این کلمه hello فقط در داکیومنت 2 وجود داشت.

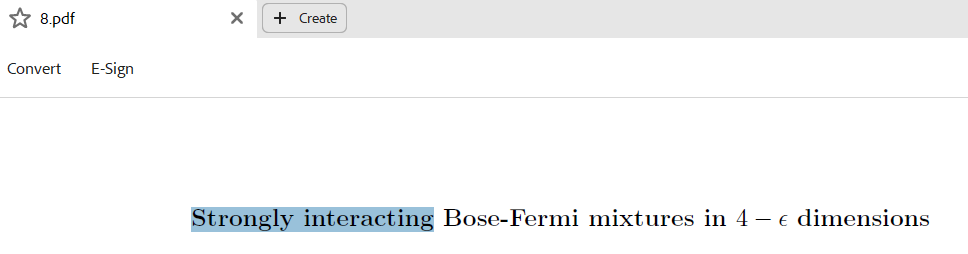
مثال بعدی:

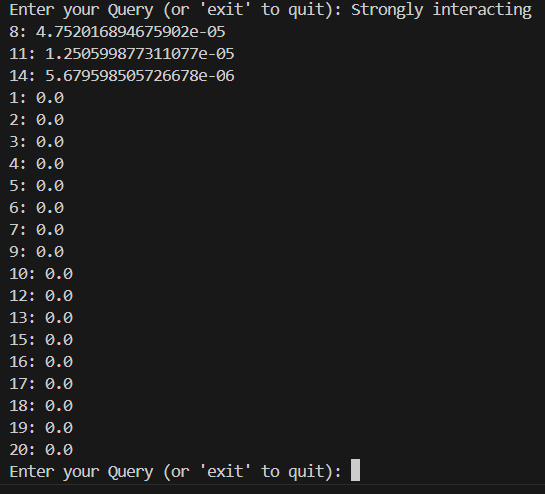




داکیومنت 10 به عنوان اولین داکیونت آورده شده است.

مثال بعدی:





داکیومنت 8 در صدر آورده شده است. مثال‌های دیگر درست جواب داده است و تست‎ها همگی پاس شدند.