



دانشکده مهندسی کامپیوتر

دانشگاه اصفهان

تکلیف چهارم درس بازیابی اطلاعات

استاد: دکتر محمد مهدی رضاپور

مهر والسادات نوحی

۹۹۳۶۱۳۰۶۱

بهار ۱۴۰۳

TF-IDF Score

صورت تمرین:

با سلام خدمت دانشجویان عزیز. پیرو طرح موضوع انجام شده در کلاس در رابطه با تکلیف جدید (شماره ۴)، شما می بایست دو سیستم بازیابی توسعه دهید:

۱. با استفاده از TF-IDF Score یک سیستم بازیابی روی اسناد جمع آوری شده در تکلیف قبل توسعه دهید.
۲. سیستم بازیابی دیگری نیز با استفاده از شاخص Cosine توسعه دهید که بر اساس وکتور وزن های TF-IDF کار کند. دقت داشته باشید که کدها می بایست با زبان پایتون نوشته شود. برنامه میبایست قابل تست باشد وگرنه نمره ای به تکلیف تعلق نمیگیرد. لذا توضیحات کافی به صورت تصویری در رابطه با برنامه توسعه داده شده در قالب یک فایل ورد پیوست تکلیف باشد. کمتر ➡

مانند گذشته تمرین را در چند گام تقسیم بندی کرده و جداگانه توضیح خواهیم داد. ترجیحا از همان ۲۰ مقاله دانلود شده در تمرین های قبل برای این تمرین و تمرین بعدی استفاده خواهد شد که نتایج به صورت واضح تر نشان داده شود.

گام اول:

در گام اول از پوشه ای که ۲۰ مقاله مورد نظر در آن است باید از هر ۲۰ مقاله شروع به خواندن و استخراج متن کنیم.

```
def main():  
    pdf_directory = "arxiv_orgs_pdfs"  
    all_texts = read_pdf(pdf_directory)
```

```
def read_pdf(pdf_directory):  
    all_texts = {}  
    try:  
        pdf_files = [filename for filename in os.listdir(pdf_directory) if filename.endswith('.pdf')]  
        pdf_files.sort(key=lambda x: int(os.path.splitext(x)[0]))  
        for filename in pdf_files:  
            pdf_path = os.path.join(pdf_directory, filename)  
            text = extract_text_from_pdf(pdf_path)  
            tokens = tokenize_text(text)  
            filename_without_extension = os.path.splitext(filename)[0]  
            all_texts[filename_without_extension] = tokens  
    except Exception as e:  
        print(f"Error reading PDFs: {e}")  
    return all_texts
```

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.

```
import os
import PyPDF2
import string
import math
from collections import Counter

def extract_text_from_pdf(pdf_path):
    text = ""
    try:
        with open(pdf_path, 'rb') as f:
            pdf_reader = PyPDF2.PdfReader(f)
            for page_number in range(len(pdf_reader.pages)):
                extracted_text = pdf_reader.pages[page_number].extract_text()
                if extracted_text:
                    text += extracted_text
    except Exception as e:
        print(f"Error extracting text from PDF: {e}")
    return text
```

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 داکيومنت و مقدار آن متن داخل فایل است .

تذکر: من اسم فایل‌ها را برای راحتی کار به مثلاً 1.pdf و.. تغییر دادم و برای 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.

```
def tokenize_text(text):
    text = text.lower()
    text = ''.join([char if char not in string.punctuation else ' ' for char in text])
    tokens = text.split()
    return tokens
```

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:

```
def build_tf_dictionary(all_texts):
    term_document_tf_dict = {}
    for doc_name, tokens in all_texts.items():
        tf_counter = Counter(tokens)
        total_tokens = len(tokens)
        tf = {term: tf_counter[term] / total_tokens for term in tf_counter}
        for term, tf_value in tf.items():
            if term not in term_document_tf_dict:
                term_document_tf_dict[term] = []
            term_document_tf_dict[term].append((doc_name, tf_value))
    return term_document_tf_dict
```

این تابع `build_tf_dictionary`، دیکشنری TF را می‌سازد. برای هر داکيومنت، فراوانی نسبی هر توکن (TF) محاسبه می‌شود و در دیکشنری ذخیره می‌شود. به ازای هر داکيومنت، برای همه توکن‌ها در تمام متن استخراج شده یک دیکشنری ساخته و به عنوان مثال کلید آن ترم یا توکن و مقدار آن مقدار `tf` آن به ازای داکيومنت، و شماره داکيومنت 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}
```

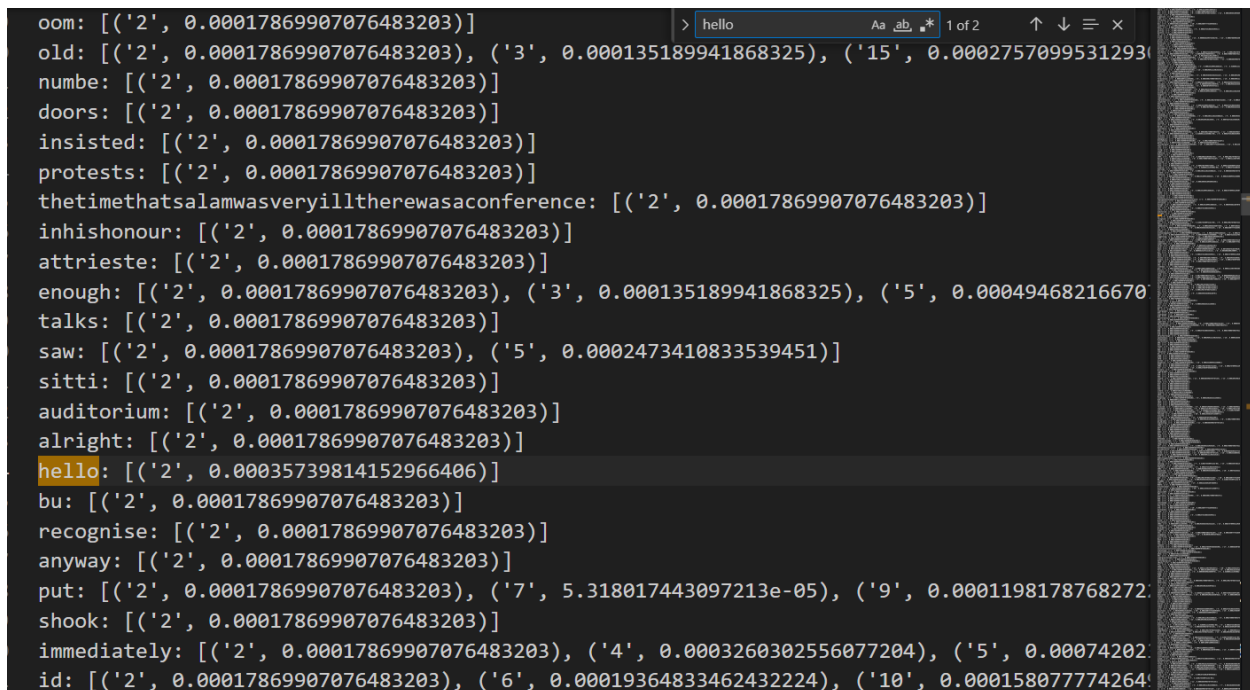
مجدد دیکشنری آپدیت می‌شود. و این نتیجه نهایی است چون پردازش در این مثال تکمیل شد.

```
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:

```
def save_tf_dictionary_to_file(term_document_tf_dict, output_file):  
    try:  
        with open(output_file, 'w', encoding='utf-8') as f:  
            for term, doc_tf_list in term_document_tf_dict.items():  
                f.write(f"{term}: {doc_tf_list}\n")  
    except Exception as e:  
        print(f"Error writing to file: {e}")
```

خروجی:



```
oom: [('2', 0.00017869907076483203)]  
old: [('2', 0.00017869907076483203), ('3', 0.000135189941868325), ('15', 0.0002757099531293)]  
numbe: [('2', 0.00017869907076483203)]  
doors: [('2', 0.00017869907076483203)]  
insisted: [('2', 0.00017869907076483203)]  
protests: [('2', 0.00017869907076483203)]  
thetimethatsalamwasveryilltherewasaconference: [('2', 0.00017869907076483203)]  
inhishonour: [('2', 0.00017869907076483203)]  
attrieste: [('2', 0.00017869907076483203)]  
enough: [('2', 0.00017869907076483203), ('3', 0.000135189941868325), ('5', 0.00049468216670)]  
talks: [('2', 0.00017869907076483203)]  
saw: [('2', 0.00017869907076483203), ('5', 0.0002473410833539451)]  
sitti: [('2', 0.00017869907076483203)]  
auditorium: [('2', 0.00017869907076483203)]  
alright: [('2', 0.00017869907076483203)]  
hello: [('2', 0.00035739814152966406)]  
bu: [('2', 0.00017869907076483203)]  
recognise: [('2', 0.00017869907076483203)]  
anyway: [('2', 0.00017869907076483203)]  
put: [('2', 0.00017869907076483203), ('7', 5.318017443097213e-05), ('9', 0.0001198178768272)]  
shook: [('2', 0.00017869907076483203)]  
immediately: [('2', 0.00017869907076483203), ('4', 0.0003260302556077204), ('5', 0.00074202)]  
id: [('2', 0.00017869907076483203), ('6', 0.00019364833462432224), ('10', 0.000158077774264)]
```

گام چهارم:

در برنامه یک دیکشنری دیگری تعریف کردم که در آن به ازای هر ترم از هر داکيومنت، 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.

$$idf_t = \log_{10} (N/df_t)$$

```
def calculate_idf(all_texts):
    total_documents = len(all_texts)
    term_document_frequency = {}
    for doc_tokens in all_texts.values():
        unique_tokens = set(doc_tokens)
        for term in unique_tokens:
            if term not in term_document_frequency:
                term_document_frequency[term] = 0
            term_document_frequency[term] += 1

    idf_dict = {}
    for term, document_frequency in term_document_frequency.items():
        if(document_frequency==0):
            idf = math.log10(total_documents / (1 + document_frequency))
        else:
            idf = math.log10(total_documents / (document_frequency))
        idf_dict[term] = idf

    return idf_dict
```

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

```
def save_idf_dictionary_to_file(idf_dict, output_file):
    try:
        with open(output_file, 'w', encoding='utf-8') as f:
            for term, idf in idf_dict.items():
                f.write(f"{term}: {idf}\n")
    except Exception as e:
        print(f"Error writing to file: {e}")
```


خروجی:

```
3877 realisation: 0.8239087409443188
3878 orking: 1.0
3879 mensional: 0.8239087409443188
3880 noether: 1.0
3881 afterwards: 0.8239087409443188
3882 pale: 1.0
3883 cle: 0.6989700043360189
3884 examination: 1.0
3885 theoretical: 0.2596373105057561
3886 professors: 0.8239087409443188
3887 prize: 1.0
3888 claiming: 1.0
3889 spontaneously: 0.8239087409443188
3890 articl: 1.0
3891 voice: 1.0
3892 hello: 1.0
3893 spacetimes: 0.6989700043360189
3894 google: 0.8239087409443188
3895 note: 0.18708664335714445
3896 resi: 1.0
3897 endeavour: 1.0
3898 unless: 0.6020599913279624
```

گام پنجم:

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

Score for a document given a query

$$\text{Score}(q, d) = \sum_{t \in q \cap d} \text{tf.idf}_{t,d}$$

پس از این رو دو تابع تعریف می‌کنیم یکی برای محاسبه tf.idf خوده کویری و یکی tf.idf کلی که حاصل ضرب این دو برای هر ۲۰ مقاله محاسبه شده و بالاترین مقدار از لحاظ رنک نشان داده می‌شود.

```
def calculate_tfidf_query(idf_dict, word, query):
    if word in idf_dict:
        idf = idf_dict[word]
        tf_word = query.count(word) / len(query)
        tfidf_query = tf_word * idf
        return tfidf_query
    else:
        return 0
```

```
def calculate_tf_idf(idf_dict, term_document_tf_dict, word, doc_name):
    tf_idf_scores = 0
    if word in idf_dict and word in term_document_tf_dict:
        idf = idf_dict[word]
        for doc, tf in term_document_tf_dict[word]:
            if doc == doc_name:
                tf_idf_scores = tf * idf
                break
        return tf_idf_scores
    else:
        return 0
```

کاری که من کردم چون بردارها را داریم اومدم و برای هر ۲۰ تا مقاله دونه دونه یک حلقه بیرونی واسه مقاله و یک حلقه درونی به ازای هر توکن در کویری این مقدار محاسبه و با تعریف یک دیکشنری جدید به نام total_score با کلید داکيومنت، و مقدار tf.idf کل واسه اون مقاله به ازای کویری 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

```

while True:
    total_score = {}
    query = input("Enter your Query (or 'exit' to quit): ")
    if query.lower() == 'exit':
        break

    query_tokens = list(set(query.lower().split()))

    for doc_name, tokens in all_texts.items():
        score_word = 0
        for term in query_tokens:
            tfidf_query = calculate_tfidf_query(idf_dict, term, query)
            tf_idf_term = calculate_tf_idf(idf_dict, term_document_tf_dict, term, doc_name)
            score_word += tf_idf_term * tfidf_query
        total_score[doc_name] = score_word

    sorted_total_score = sorted(total_score.items(), key=lambda item: item[1], reverse=True)
    scaling_factor = 1e6

    for doc, score in sorted_total_score:
        scaled_score = score * scaling_factor
        print(f"{doc}: {score}")

```

گام ششم:

در این گام میریم سراغ تست برنامه :

```

PS E:\Computer_Engineering\08\Data_Retrieval\Hws\tf_idf> & C:/Users/mahroonoohi/ineering/08/Data_Retrieval/Hws/tf_idf/main.py
Enter your Query (or 'exit' to quit): hello
2: 0.00012099206930789271
1: 0.0
3: 0.0
4: 0.0
5: 0.0
6: 0.0
7: 0.0
8: 0.0
9: 0.0
10: 0.0
11: 0.0
12: 0.0
13: 0.0
14: 0.0
15: 0.0
16: 0.0
17: 0.0
18: 0.0
19: 0.0
20: 0.0
Enter your Query (or 'exit' to quit): 

```

رفتم چک کردم و این کلمه hello فقط در داکيومنت ۲ وجود داشت.

مثال بعدی:



What an event is not: unravelling the identity of events in quantum theory and gravity

Anne-Catherine de la Hamette^{*,†}, Viktoria Kabel[†], and Časlav Brukner
*University of Vienna, Faculty of Physics, Vienna Doctoral School in Physics,
and Vienna Center for Quantum Science and Technology (VCQ),
Boltzmannngasse 5, A-1090 Vienna, Austria and
Institute for Quantum Optics and Quantum Information (IQOQI),
Austrian Academy of Sciences, Boltzmannngasse 3, A-1090 Vienna, Austria*

```
Enter your Query (or 'exit' to quit): What an event is not
10: 0.00022782947321157258
20: 4.192740330070751e-05
18: 5.6968174785622e-06
14: 4.087676263064711e-06
1: 0.0
2: 0.0
3: 0.0
4: 0.0
5: 0.0
6: 0.0
7: 0.0
8: 0.0
9: 0.0
11: 0.0
12: 0.0
13: 0.0
15: 0.0
16: 0.0
17: 0.0
19: 0.0
Enter your Query (or 'exit' to quit):
```

داکیومنت ۱۰ به عنوان اولین داکيومنت آورده شده است.

مثال بعدی:

☆ 8.pdf



+ Create

Convert

E-Sign

Strongly interacting Bose-Fermi mixtures in $4 - \epsilon$ dimensions

```
Enter your Query (or 'exit' to quit): Strongly interacting
8: 4.752016894675902e-05
11: 1.250599877311077e-05
14: 5.679598505726678e-06
1: 0.0
2: 0.0
3: 0.0
4: 0.0
5: 0.0
6: 0.0
7: 0.0
9: 0.0
10: 0.0
12: 0.0
13: 0.0
15: 0.0
16: 0.0
17: 0.0
18: 0.0
19: 0.0
20: 0.0
Enter your Query (or 'exit' to quit):
```

داکیومنت ۸ در صدر آورده شده است. مثال‌های دیگر درست جواب داده است و تست‌ها همگی پاس شدند.