



دانشکده مهندسی کامپیوتر

دانشگاه اصفهان

تکلیف پنجم (اختیاری و امتیازی) درس بازیابی اطلاعات

استاد: دکتر محمد مهدی رضاپور

مهروالسادات نوحی

۹۹۳۶۱۳۰۶۱

بهار ۱۴۰۳

صورت تمرین:

با سلام خدمت دانشجویان عزیز. پیرو طرح موضوع انجام شده در کلاس در رابطه با تکلیف تشویقی جدید (شماره ۵)، شما می بایست با استفاده از روش BM25 score یک سیستم بازیابی روی اسناد جمع آوری شده در تکالیف قبلی توسعه دهید. همچنین در گزارش تکلیف تاثیر ایجاد تغییر در فاکتورهای k و b را روی نتایج نشان داده شده سیستم بررسی کنید.

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 خواهم داد. ترجیحا از همان ۲۰ مقاله دانلود شده در تمرینهای 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 پوشه‌ای 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

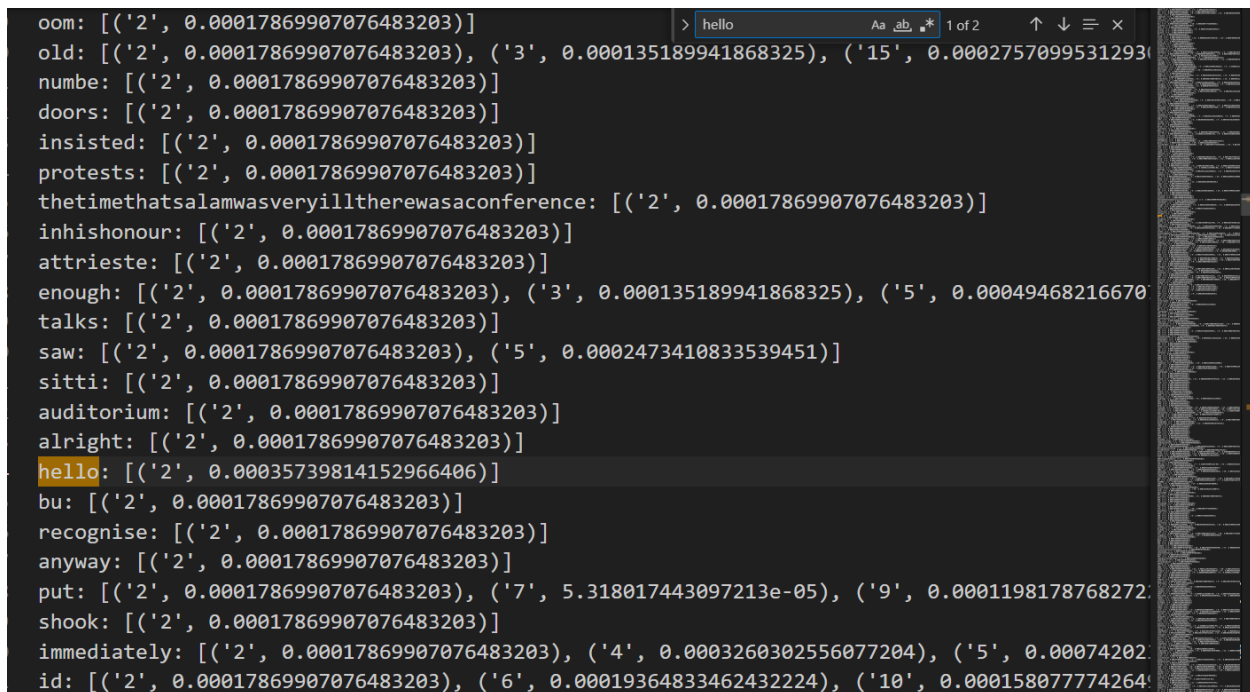
Term2 → Document_ID → tf #for that document find tf

...}

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', 0.00015807774264)]
```

گام چهارم:

در برنامه یک دیکشنری دیگری تعریف کردم که در آن به ازای هر ترم ازهر 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.

$$\text{idf}_t = \log_{10} (N/\text{df}_t)$$

```
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():
        #idf = math.log10((total_documents - document_frequency + 0.5) / (document_frequency + 0.5) + 1)
        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:

در این گام بپردازیم به BM25 چطور کار می کند:

$$score(q, d) = \sum_{i=1}^{|q|} idf(q_i) \cdot \frac{tf(q_i, d) \cdot (k_1 + 1)}{tf(q_i, d) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avgdl})}$$

$$TF \text{ component} = \frac{tf \cdot (k_1 + 1)}{tf + k_1 \cdot (1 - b + b \cdot (\frac{doc_length}{avgdl}))}$$

$$IDF(t) = \log \left(\frac{N - n(t) + 0.5}{n(t) + 0.5} + 1 \right)$$

بریم برنامه رو ببینیم چطوری کار می کند:

پارامترها

۱. **idf_dict**: یک دیکشنری که کلمات (ترمها) را به عنوان کلیدها دارد و مقادیرشان فراوانی معکوس سند است. فراوانی معکوس سند اهمیت یک کلمه را در کل مجموعه اسناد اندازه گیری می کند.

۲. **term_document_tf_dict**: یک دیکشنری که کلمات (ترمها) را به عنوان کلیدها دارد و مقادیرش لیستی از تاپل ها هستند. هر تاپل شامل:

- نام سند
- فراوانی کلمه (TF) در آن سند
- طول سند

۳. **query_tokens**: یک لیست از کلمات (ترم‌ها) از پرسمان.

۴. **doc_name**: نام سندی که امتیاز BM25 برای آن محاسبه می‌شود.

۵. **k**: یک پارامتر تنظیم کننده که مقیاس فراوانی کلمات را تنظیم می‌کند. پیش فرض آن ۱.۵ است.

۶. **b**: یک پارامتر تنظیم کننده که تأثیر نرمال سازی طول سند را کنترل می‌کند. پیش فرض آن ۰.۷۵ است.

۷. **avgdl**: میانگین طول سند در کل مجموعه اسناد. این مقدار برای نرمال سازی طول استفاده می‌شود.

```
def bm25_score(idf_dict, term_document_tf_dict, query_tokens, doc_name, k1=1.5, b=0.75, avgdl=0):
    score = 0.0
    for term in query_tokens:
        if term in idf_dict and term in term_document_tf_dict:
            idf = idf_dict[term]
            for doc, tf, doc_length in term_document_tf_dict[term]:
                if doc == doc_name:
                    tf_component = ((tf * (k1 + 1)) / (tf + k1 * (1 - b + b * (doc_length / avgdl))))
                    score += idf * tf_component
                    break
    return score
```

```
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 = bm25_score(idf_dict, term_document_tf_dict, query_tokens, doc_name, avgdl=avgdl)
        total_score[doc_name] = score

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

    for doc, score in sorted_total_score:
        print(f"{doc}: {score}")
```

گام ششم:

بریم چند تا تست کنیم والبته مقدار k, b هم تغییر بدیم تا تغییرات را بهتر ببینیم.

با $k=1.5$ و $b=0.75$

```
PS E:\Computer_Engineering\08\Data_Revtrial\Hws\BM25> & C:/Users/mahroonoohi/Computer_Engineering/08/Data_Revtrial/Hws/BM25/main.py
Enter your Query (or 'exit' to quit): hello
2: 1.986586158122443
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):
```

در بخش‌های قبل نیز داکيومنت ۲ برگردانده شد.

☆ 10.pdf

×

+ Create

Convert E-Sign

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

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

```
Enter your Query (or 'exit' to quit): what an event is not
10: 2.4027719870017776
20: 2.0138820944802163
18: 1.2110755983477626
14: 0.7481630104392127
2: 0.68537948548715
11: 0.5353993795793222
7: 0.5066540775533586
3: 0.4946767085678854
16: 0.3927036882163906
8: 0.3281331179285727
9: 0.27702931225585753
1: 0.0
4: 0.0
5: 0.0
6: 0.0
12: 0.0
13: 0.0
15: 0.0
17: 0.0
19: 0.0
Enter your Query (or 'exit' to quit):
```

داکیومنت ۱۰ به عنوان اولین داکایونت آورده شده است.

```
Enter your Query (or 'exit' to quit): information
19: 0.30884139966830754
11: 0.2988763860868461
14: 0.2832461199971051
17: 0.28265749247941035
12: 0.27456338363359317
13: 0.2658729867309634
10: 0.26054083032158404
4: 0.2596675739397107
6: 0.2400316352367757
1: 0.23054818058283655
20: 0.22720324748912513
16: 0.21420071654338915
15: 0.20634865663399338
9: 0.16808249649461676
7: 0.07102543345494632
2: 0.0
3: 0.0
5: 0.0
8: 0.0
18: 0.0
Enter your Query (or 'exit' to quit):
```

بریم با همین مثال فاکتورهای k, b تغییر بدهیم.

این سری $k=1.8$ و $b=0.75$ است:

```
PS E:\Computer_Engineering\08\Data_Retrieval\Hws\BM25> & C:/Users/mahroonoohi/Computer_Engineering/08/Data_Retrieval/Hws/BM25/main.py
Enter your Query (or 'exit' to quit): hello
2: 2.06428180344745
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
```

مقدار عددی عوض شده ولی همچنان داکيومنت ۲ که درست است برگردانده شده است.

این سری $k=1.8$ و $b=1$ است: مقدار عددی عوض شده ولی همچنان داکيومنت ۲ که درست است برگردانده شده است.

```
89
90 > def bm25_score(idf_dict, term_document_tf_dict, query_tokens, doc_name, k1=1.8, b=1, avgdl=0): ...
101
102 def main():
103     pdf_directory = "arxiv_orgs_pdfs"
104     all_texts = read_pdf(pdf_directory)

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\Computer_Engineering\08\Data_Retrieval\Hws\BM25> & C:/Users/mahroonoohi/AppData/Local/Programs/Python/Python312/python.exe e:/Computer_Engineering/08/Data_Retrieval/Hws/BM25/main.py
Enter your Query (or 'exit' to quit): hello
2: 2.11841195491741
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):
```

این سری $k=1.8$ و $b=0$ است:

```
89
90 def bm25_score(idf_dict, term_document_tf_dict, query_tokens, doc_name, k1=1.8, b=0, avgdl=0):
91     score = 0.0

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\Computer_Engineering\08\Data_Retrieval\Hws\BM25> & C:/Users/mahroonoochi/AppData/Local/Programs/Python/Python312/python.exe e:\Computer_Engineering\08\Data_Retrieval\Hws\BM25/main.py
Enter your Query (or 'exit' to quit): hello
2: 1.91730736203113
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):
```

با عوض شدن مقدار پارامترها عدد خروجی هم تغییر کرده و داکيومنت ۲ همچنان برگردانده شده است. من برای جمله هم تست کردم و اوکی بوده است.

```
89
90 def bm25_score(idf_dict, term_document_tf_dict, query_tokens, doc_name, k1=1.8, b=1, avgdl=0):
91     score = 0.0

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS E:\Computer_Engineering\08\Data_Retrieval\Hws\BM25> & C:/Users/mahroonoochi/AppData/Local/Programs/Python/Python312/python.exe e:\Computer_Engineering\08\Data_Retrieval\Hws\BM25/main.py
Enter your Query (or 'exit' to quit): i love me
2: 1.795595540539839
11: 1.1525835273707883
20: 0.8900907256484736
7: 0.33533612079011726
1: 0.0
3: 0.0
4: 0.0
5: 0.0
6: 0.0
8: 0.0
9: 0.0
10: 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
Enter your Query (or 'exit' to quit):
```