به نام خدا

تمرین ۱

درس بازیابی اطلاعات

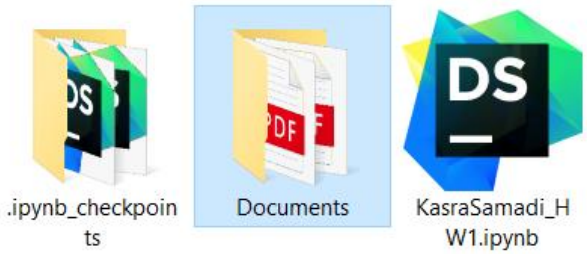
استاد : دکتر رضاپور

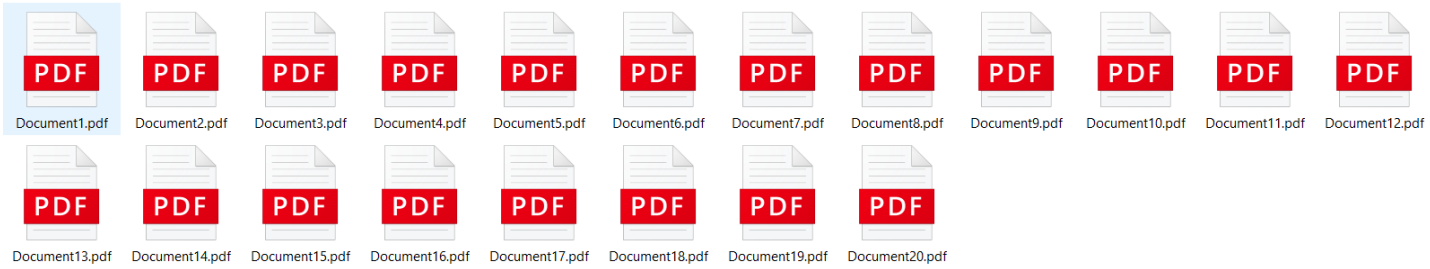
کسری صمدی<۹۹۳۶۲۳۰۳۰>

بهمن ۱۴۰۲

شما می‌بایست یک ایندکس معکوس روی ۲۰ مقاله از پایگاه arxiv.org که به صورت رندوم و البته در موضوعات متنوع دانلود کرده‌اید ایجاد کنید. سپس می‌بایست برنامه‌ای بنویسید که کوئری‌های بولینی کاربر را توسط این ایندکس پاسخ دهد. کدها می‌بایست با زبان پایتون نوشته شود. برنامه می‌بایست قابل تست باشد وگرنه نمره‌ای 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.

همانطور که در این تمرین خواسته شده است، ابتدا ۲۰ مقاله از سایت مورد نظر با فرمت pdf دانلود و به ترتیب با نام‌های Document1 تا Document20 نام‌گذاری شد و سپس تمامی این مقالات در پوشه‌ای با نام Documents در کنار فایل اجرایی کد پایتون ذخیره شده است.





حال به سراغ بخش کد پایتون می‌رویم :

در این بخش ابتدا کتابخانه‌های موردنیاز را ایمپورت می‌کنیم.

```
In [1]: import nltk
        import PyPDF2
        import string
        from ordered_set import OrderedSet
```

کتابخانه nltk : برای پردازش زبان طبیعی استفاده می‌شود و در این تمرین، برای tokenize کردن جملات موجود در مقالات استفاده شده است.

کتابخانه pyPDF2 : برای استخراج متن مقالات کاربرد دارد.

کتابخانه string : برای کار با دیتاتایپ استرینگ استفاده می‌شود و در این تمرین، برای حذف علائم نگارشی (Remove punctuation) استفاده شده است.

کتابخانه ordered_set : یک داده ساختار ست (مجموعه) است که عناصر با ترتیب در آن ذخیره می‌شوند. همانطور که می‌دانید در مجموعه‌ها (Set)، عناصر تکراری ذخیره نمی‌شوند.

```
In [2]: documents_folder_name = "Documents"
        documents_name = "Document"
        documents_count = 20

        inverted_index = {}
```

حال در این قسمت، متغیرهای ثابت را تعریف می‌کنیم که به ترتیب شامل نام فولدر ذخیره کننده مقالات (Documents)، نام ثابت برای هر مقاله (Document) و تعداد مقالات (۲۰) است.

همچنین یک دیکشنری که ایندکس‌های معکوس را در خود ذخیره می‌کند، تعریف شده است. کلیدهای این دیکشنری شامل کلمات استخراج شده از مقالات است و مقادیر (value) این دیکشنری شامل مجموعه‌ای (set) از نام‌های مقالاتی است که کلمه موردنظر در آن مقالات وجود دارد.

```
In [3]: for i in range(documents_count):
            doc_name = f"{documents_name}{i + 1}"
            doc_address = fr"{documents_folder_name}\{doc_name}.pdf"

            with open(doc_address, "rb") as pdf_file:
                read_pdf = PyPDF2.PdfReader(pdf_file)
                number_of_pages = len(read_pdf.pages)
```

در این قسمت، نام‌های مقالات (Document1 تا Document20) همچنین آدرس آن‌ها (\Documents\Document1..20) به ازای هر مقاله ساخته می‌شود و با استفاده از کتابخانه PyPDF2 مقاله موردنظر خوانده شده و تعداد صفحات آن بدست می‌آید.

```
number_of_pages = len(read_pdf.pages)
for page_num in range(number_of_pages):
    page = read_pdf.pages[page_num]
    page_content = page.extract_text()

    lower_text = page_content.lower()
    text_without_punctuation_marks = lower_text.translate(str.maketrans('', '', string.punctuation))

    tokens = nltk.word_tokenize(text_without_punctuation_marks)

    for item in tokens:
        if item not in inverted_index:
            inverted_index[item] = OrderedSet()
            inverted_index[item].add(doc_name)
        if item in inverted_index:
            inverted_index[item].add(doc_name)
```

حال با استفاده از تعداد صفحات بدست آمده برای مقاله موردنظر، متن هر صفحه از آن استخراج می‌شود. سپس متن بدست آمده به حروف کوچک (lowercase) تبدیل شده و علائم نگارشی (punctuation marks) از آن حذف می‌شود.

بعد از آن با استفاده از کتابخانه nltk، متن موردنظر tokenize (کلمه کلمه) می‌شود. حال کلمات بدست آمده از متن هر صفحه، به صورت کلید دیکشنری inverted_index ذخیره می‌شود که مقدار (Value) آن، یک مجموعه (OrderedSet) است.

اگر کلمه موردنظر داخل inverted_index نباشد، آن کلمه به عنوان یک کلید جدید به دیکشنری اضافه شده و به عنوان مقدار آن، یک OrderedSet قرار داده می‌شود و در آن OrderedSet نام مقاله‌ای که کلمه موردنظر از آن استخراج شده، ذخیره می‌شود. اگر کلمه موردنظر از قبل داخل کلیدهای دیکشنری inverted_index باشد، فقط نام مقاله‌ای که کلمه موردنظر از آن استخراج شده به OrderedSet آن اضافه می‌شود.

نکته ۱ : از مجموعه یا Set به این علت استفاده شده است که نام مقالاتی که کلماتی تکراری در آن‌ها وجود دارند، فقط یک بار به مجموعه اضافه شود. برای مثال در Document1 ، ۲۰ با کلمه word1 به کار رفته است و با این کار در inverted_index با کلید word1 یک OrderedSet با یک عضو Document1 ذخیره می‌شود و مقاله Document1 فقط یکبار در OrderedSet مربوط به کلید word1 ذخیره می‌شود.

نکته ۲ : از OrderedSet به این علت استفاده شده است که نام‌های مقالات به ترتیب در مجموعه ذخیره شوند.

```
In [4]: for term, documents in inverted_index.items():
            print(term, "->", ", ".join(documents))
```

حال inverted_index ساخته شده را با قطعه کد بالا نمایش می‌دهیم که بخشی از خروجی آن مطابق تصویر زیر است:

```
efficient -> Document1, Document3, Document4, Document5, Document8, Document13, Document14, Document15
video -> Document1, Document2, Document3, Document6
object -> Document1, Document3, Document6, Document8, Document9, Document11
segmentation -> Document1, Document4, Document5, Document6, Document7
via -> Document1, Document3, Document4, Document5, Document8, Document9, Document11, Document13, Document15, Document19
modulated -> Document1, Document9
crossattention -> Document1, Document3
memory -> Document1, Document4, Document5, Document8, Document11, Document13
abdelrahman -> Document1
shaker1 -> Document1
syed -> Document1, Document5
talal1 -> Document1
martin -> Document1, Document5, Document7, Document8, Document9
danelljan2 -> Document1
salman -> Document1, Document5, Document13
khan1 -> Document1
minghsuan -> Document1, Document5
yang345 -> Document1
```

همانطور که در تصویر بالا مشخص است، کلمات استخراج شده به همراه نام مقالات آن‌ها به‌دست آمده است.

نام مقالات با ترتیب و به صورت صعودی ذخیره شده‌اند و همچنین نام تکراری در آن‌ها وجود ندارد.

```
ans1 = int(input("(1) Single word Search\n(2) Combined search\n "))

if ans1 == 1 : # Single word Search
    word = input("Write your word to search : (Input must be in lowercase letters) ")
    res = inverted_index.get(word)
    if res is not None:
        res_dict = dict({word: res})
        for term, documents in res_dict.items():
            print(term, "->", ", ".join(documents))
    else :
        print(f"\'{word}\' NOT FOUND")
```

حال نوبت ورودی گرفتن از کاربر و تست برنامه است. در اینجا کاربر می‌تواند دو نوع جستوجو را انجام دهد:

۱- جستوجو تک کلمه‌ای

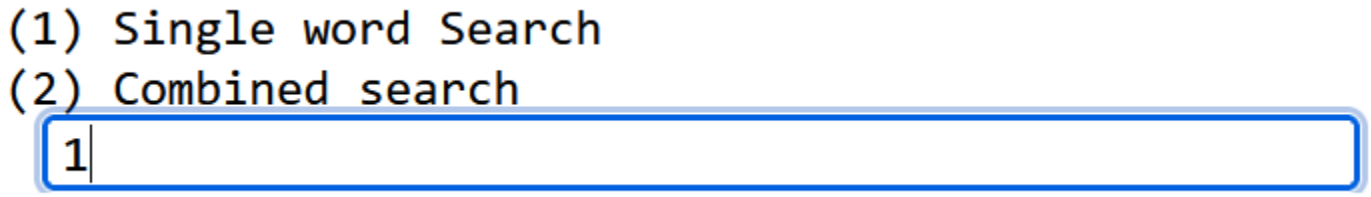
۲- جستوجو ترکیبی

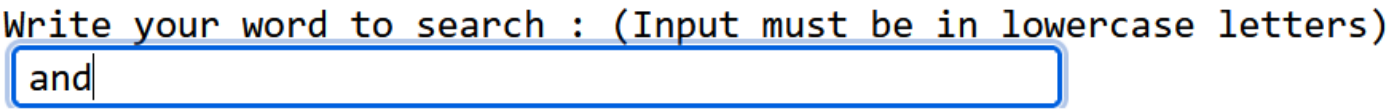
کاربر با وارد کردن شماره‌ی جستوجوی موردنظر خود، می‌تواند جستوجو را انجام دهد.

۱- جستوجو تک کلمه‌ای

در جستوجو تک کلمه‌ای کاربر فقط یک تک کلمه را وارد می‌کند و مقالاتی که شامل آن کلمه هستند به او نمایش داده می‌شود و در صورت یافت نشدن کلمه‌ی موردنظر در لیست کلیدهای دیکشنری inverted_index، پیغام "'word' NOT FOUND" نمایش داده می‌شود.

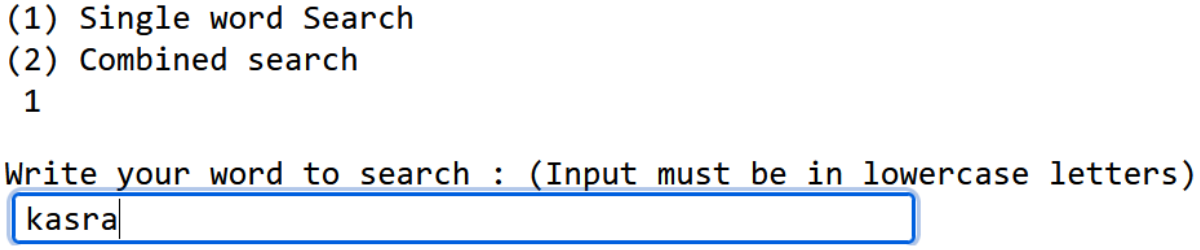
نمونه ورودی‌های این بخش در زیر آمده است :






```
Write your word to search : (Input must be in lowercase letters) and
and -> Document1, Document2, Document3, Document4, Document5, Document6, Document7, Document8, Docume
nt9, Document10, Document11, Document12, Document13, Document14, Document15, Document16, Document17,
Document18, Document19, Document20
```

کلمه‌ی and در inverted_index یافت شد و لیستی از نام مقالاتی که در آن‌ها، این کلمه وجود دارد، نمایش داده شده است.




```
Write your word to search : (Input must be in lowercase letters) kasra
'kasra' NOT FOUND
```

کلمه kasra در inverted_index یافت نشد و پیغام 'kasra' NOT FOUND چاپ شده است.

۲- جستوجو ترکیبی

```
elif ans1 == 2 : # Combined search
    print("(1) AND")
    print("(2) OR")
    op = int(input("Enter your Operator : "))
    word1, word2 = input("Write your Two words to search (Separated by space) : ").split(" ")
```

در جستوجوی ترکیبی، کاربر می‌تواند دو نوع جستوجو را انجام دهد:

۱- جستوجو ترکیبی AND

۲- جستوجو ترکیبی OR

کاربر با وارد کردن شماره‌ی جستوجوی موردنظر خود، می‌تواند جستوجو را انجام دهد.

بعد از مشخص شدن نوع جستوجو، کاربر باید دو کلمه‌ای را که می‌خواهد به صورت AND یا OR جستوجو کند، با فاصله (space) از هم وارد کند.

۱- جستوجو ترکیبی AND

```
  if op == 1:  # AND
      if word1 in inverted_index.keys() and word2 in inverted_index.keys() :
          res1 = inverted_index.get(word1)
          res2 = inverted_index.get(word2)
          and_result = res1.intersection(res2)

          if len(and_result) > 0:
              and_result_dict = dict({f"{word1} & {word2}": and_result})
              for term, documents in and_result_dict.items():
                  print(term, "->", ", ".join(documents))
          else:
              print(f"There is No document intersection between \'{word1}\' and \'{word2}\'")
      else :
          print(f"\'{word1}\' or \'{word2}\' NOT FOUND")
```

کاربر در این نوع جستوجو، دو کلمه‌ای را که می‌خواهد به صورت AND جستوجو شوند، با فاصله از هم وارد می‌کند و سپس لیستی از مقالاتی که شامل توامان آن دو کلمه هستند، نمایش داده می‌شود. اگر کلمه ۱ یا کلمه ۲ در دیکشنری inverted_index نباشند، پیغام NOT FOUND چاپ می‌شود و اگر دو کلمه‌ی موردنظر کاربر، اشتراکی از لحاظ نام مقالاتی که شامل آن‌ها هستند، نداشته باشند، پیغام "'\There is No document intersection between \'{word1}\' and \'{word2}" نمایش داده می‌شود.

نمونه ورودی‌های این بخش در زیر آمده است :

```
(1) Single word Search
(2) Combined search
2

(1) AND
(2) OR

Enter your Operator : 1

Write your Two words to search (Separated by space) :
video object

Write your Two words to search (Separated by space) : video object
video & object -> Document1, Document3, Document6
```

به عنوان ورودی، دو کلمه‌ی video و object وارد شد و به عنوان خروجی، لیستی از نام‌های مقالاتی که شامل این دو کلمه به صورت and هستند، نمایش داده شده است.

۲- جستوجو ترکیبی OR

```
    elif op == 2:  # OR
        if (word1 in inverted_index.keys() or word2 in inverted_index.keys()):
            res1 = inverted_index.get(word1)
            res2 = inverted_index.get(word2)

            if not res1:  # Check if res1 is empty
                or_result = res2
                or_result_dict = dict({f"{word2}": res2})
            elif not res2:  # Check if res2 is empty
                or_result = res1
                or_result_dict = dict({f"{word1}": res1})
            else:
                or_result = res1.union(res2)
                or_result_dict = dict({f"{word1} | {word2}": or_result})

            for term, documents in or_result_dict.items():
                print(term, "->", ", ".join(documents))

        else :
            print(f"\'{word1}\' and \'{word2}\' NOT FOUND")
    else:
        print("Invalid Input")
else:
    print("Invalid Input")
```

کاربر در این نوع جستوجو، دو کلمه‌ای را که می‌خواهد به صورت OR جستوجو شوند، با فاصله از هم وارد می‌کند و سپس لیستی از مقالاتی که شامل آن دو کلمه به صورت OR هستند، نمایش داده می‌شود. اگر کلمه ۱ و کلمه ۲ در دیکشنری inverted_index نباشند، پیغام NOT FOUND چاپ می‌شود. اگر کلمه ۱ در inverted_index نباشد، به عنوان نتیجه، لیست نام‌های مقالاتی که کلمه ۲ در آن‌ها وجود دارد، چاپ می‌شود و همچنین اگر کلمه ۲ در inverted_index نباشد، به عنوان نتیجه، لیست نام‌های مقالاتی که کلمه ۱ در آن‌ها وجود دارد، چاپ می‌شود و اگر هر دو کلمه در inverted_index باشد، اجتماع یا OR لیست مقالاتی که شامل آن دو کلمه هستند، چاپ می‌شود.

نمونه ورودی‌های این بخش در زیر آمده است :

```
(1) Single word Search
(2) Combined search
 2
(1) AND
(2) OR

Enter your Operator : 2
```

```
Write your Two words to search (Separated by space) :
video object
```

```
Write your Two words to search (Separated by space) : video object
video | object -> Document1, Document2, Document3, Document6, Document8, Document9, Document11
```

به عنوان ورودی، دو کلمه‌ی video و object وارد شد و به عنوان خروجی، لیستی از نام‌های مقالاتی که شامل این دو کلمه به صورت or هستند، نمایش داده شده است.

```
(1) Single word Search
(2) Combined search
 2
(1) AND
(2) OR
Enter your Operator : 2

Write your Two words to search (Separated by space) :
video kasra
```

```
Write your Two words to search (Separated by space) : video kasra
video -> Document1, Document2, Document3, Document6
```

چون کلمه kasra در inverted_index وجود ندارد، فقط لیستی از مقالاتی که کلمه video در آن‌ها وجود دارد نمایش داده می‌شود.