



دانشکده مهندسی کامپیوتر

دانشگاه اصفهان

تکلیف اول بازیابی اطلاعات

استاد: دکتر محمد مهدی رضاپور

مهر والسادات نوحی

۹۹۳۶۱۳۰۶۱

بهار ۱۴۰۳

ایندکس معکوس:

صورت تمرین: 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 و گرنه نمره‌ای به تکلیف تعلق نمی‌گیرد.

گام اول:

در این گام ابتدا با مراجعه به arxiv.org شروع به انتخاب و دانلود ۲۰ مقاله کردم و در پوشه `arxiv_orgs_pdfs` فایل‌ها را قرار دادم. و در گام بعد شروع به کدنویسی می‌کنیم.

Name	Date modified	Type	Size
.idea	4/2/2024 12:02 PM	File folder	
arxiv_orgs_pdfs	4/2/2024 1:13 AM	File folder	
inverted_index.txt	4/3/2024 10:55 AM	Text Document	365 KB
main.py	4/2/2024 10:50 AM	JetBrains PyCharm ...	4 KB

گام دوم:

در این گام باید متن‌های داخل فایل‌های `pdf` را استخراج کنیم که بتوانیم روی آن‌ها کار کنیم. برای این کار دو کتابخانه معروف پایتون هست. ۱- `pdfpy2` و ۲- `pdfplumber` که من از اولی استفاده کردم. و خروجی برنامه را در `inverted_index.txt` ریخته می‌شود. با فراخوانی تابع `read_pdf` و دادن آدرس مکان `pdfs` ها شروع به استخراج متن می‌کنیم.

```
def main():  
    pdf_directory = "arxiv_orgs_pdfs"  
    output_file = "inverted_index.txt"  
    all_texts = read_pdf(pdf_directory)
```

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.

```
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)
            filename_without_extension = os.path.splitext(filename)[0]
            all_texts.append((filename_without_extension, text)) # Store filename without extension as
    except Exception as e:
        print(f"Error reading PDFs: {e}")
    return all_texts
```

```
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)):
                text += pdf_reader.pages[page_number].extract_text()
    except Exception as e:
        print(f"Error extracting text from PDF: {e}")
    return text
```

در واقع به شکل زیرشده است:

```
{
  "Filename1": text1,
  "Filename2": text2,
  ...}
```

گام سوم:

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):
    # Convert text to lowercase
    text = text.lower()
    # Remove punctuation
    text = ''.join([char if char not in string.punctuation else ' ' for char in text])
    # Split text into words
    tokens = text.split()
    return tokens
```

گام چهارم:

در این گام شروع به ایجاد جدول معکوس می‌کنیم:

```
def create_inverted_index(all_texts):
    inverted_index = {}
    try:
        for filename, text in all_texts:
            tokens = tokenize_text(text)
            for token in tokens:
                if token not in inverted_index:
                    inverted_index[token] = []
                if filename not in inverted_index[token]:
                    inverted_index[token].append(filename)
    except Exception as e:
        print(f"Error creating inverted index: {e}")
    return inverted_index
```

حلقه بیرونی برای پیمایش هر داکيومنت و حلقه داخلی به ازای هر توکن در متن چک می‌کند اگر توکن در جدول که به صورت دیکشنری تعریف شده است که کلید آن توکن یا کلمه هست و مقدار آن لیستی از داکيومنت یا داکيومنت‌هایی که آن کلمه در فایل‌ها موجود است اگر وجود نداشته باشد به جدول اضافه می‌شود و روبرو آن یک لیست خالی تعریف شده و اگر شماره فایل نبود آن فایل به لیست اضافه می‌شود.

گام پنجم:

برای نوشتن جدول معکوس در فایلی که در ابتدا گفتیم این تابع را تعریف می‌کنیم:

```
def print_inverted_index_to_file(inverted_index, output_file):
    try:
        with open(output_file, "w", encoding="utf-8") as f:
            for word, docs in inverted_index.items():
                f.write(f"{word}: {docs}\n")
            print(f"Inverted index saved to {output_file}")
    except Exception as e:
        print(f"Error printing inverted index to file: {e}")
```

گام ششم:

جدول ایجاد و آماده سرچ کاربر می‌باشد (بستگی دارد صرفاً سرچ کاربر یک کلمه ساده یا کویری بولینی باشد تصمیم می‌گیریم چی در خروجی به او نشان دهیم).

```
def search_inverted_index(inverted_index, query):
    terms = query.lower().split()
    result = set()
    if terms[0] in inverted_index:
        result.update(inverted_index[terms[0]])

    for i in range(1, len(terms), 2):
        operator = terms[i]
        next_term = terms[i + 1] if i + 1 < len(terms) else None

        if operator == "and" and next_term:
            if next_term in inverted_index:
                result.intersection_update(inverted_index[next_term])

        elif operator == "or" and next_term:
            if next_term in inverted_index:
                result.update(inverted_index[next_term])

        elif operator == "not" and next_term:
            if next_term in inverted_index:
                result.difference_update(inverted_index[next_term])

    return result
```





بعد از اینکه کاربر کویری خود را وارد کرد کلمه به کلمه پردازش می‌کنیم. و اول کلمه اول را سرچ کرده که در جدول معکوس هست یا خیر اگر بود در result اضافه می‌کنیم در For میبینیم اگر کویری تک کلمه ای بوده باشد که خروجی برگردانده می‌شود و اگر کویری بولینی باشد شروع به پیدا کردن اپراتور و کلمه بعدی کرده و مطابق با اپراتور نتیجه اپدیت می‌شود.

و شروع برنامه main است که به صورت یک حلقه همیشه درست برای کویری کاربر نوشته شده است.

```
def main():
    pdf_directory = "arxiv_orgs_pdfs"
    output_file = "inverted_index.txt"
    all_texts = read_pdf(pdf_directory)
    inverted_index = create_inverted_index(all_texts)
    print_inverted_index_to_file(inverted_index, output_file)





    print("1. Search Inverted Index")
    print("2. Exit")
    while True:
        choice = input("Enter your choice (1-2): ")
        if choice == "1":
            Query = input("Enter your Query: ")
            results = search_inverted_index(inverted_index, Query)
            if results:
                for filename in results:
                    print(filename)
            else:
                print("No documents found.")
        elif choice == "2":
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")
```

حالا شروع به اجرا کرده در ابتدا فایل خالی است. بعد از اجرای برنامه جدول معکوس را خواهیم دید.

	.idea	4/2/2024 12:02 PM	File folder	
	arxiv_orgs_pdfs	4/2/2024 1:13 AM	File folder	
	inverted_index.txt	4/4/2024 12:20 AM	Text Document	0 KB
	main.py	4/2/2024 10:50 AM	JetBrains PyCharm ...	4 KB

بعد از اجرا برنامه :

```
inverted_index.py
Inverted index saved to inverted_index.txt
1. Search Inverted Index
2. Exit
Enter your choice (1-2):
```

 .idea	4/2/2024 12:02 PM	File folder	
 arxiv_orgs_pdfs	4/2/2024 1:13 AM	File folder	
 inverted_index.txt	4/4/2024 12:22 AM	Text Document	365 KB
 main.py	4/2/2024 10:50 AM	JetBrains PyCharm ...	4 KB

```

1: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20']
optical: ['1', '6', '10', '20']
screening: ['1']
of: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20']
citrus: ['1']
leaf: ['1', '13']
diseases: ['1']
using: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '20']
label: ['1', '7', '10', '11']
free: ['1', '2', '3', '4', '6', '8', '9', '12', '18', '20']
spectroscopic: ['1']
tools: ['1', '5', '11', '13', '14', '17', '20']
a: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20']
review: ['1', '2', '3', '4', '5', '6', '8', '13', '16', '19']
saurav: ['1']
bharadwaj1: ['1']
2: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20']
akshita: ['1']
midha2: ['1']
shikha: ['1']
sharma3: ['1']
gurupkar: ['1']
singh: ['1', '6', '15', '16']
sidhu4: ['1']
and: ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12', '13', '14', '15', '16', '17', '18', '19', '20']
rajesh: ['1']

```

حالا بریم چند کوئری ببینیم:

```

Enter your Query: hello
2
Enter your choice (1-2): 1
Enter your Query: this
1
8
19
20
16
18
4
2
6
10
14
17
12
5
13
3
7
11
15
9

```

کلمه hello در فایل ۲ فقط هست و کلمه this در هر ۲۰ فایل من بوده است حالا بریم کوعری بولینی هم بزنیم:

```
Enter your Query: this not hello
1
8
19
20
16
18
4
6
10
14
17
12
5
13
3
7
11
15
9
```

الان شماره فایل ۲ دیگر نیست چون از not استفاده شده است. من تست‌های دیگری هم زدم و اوکی بوده است.

```
Enter your Query: good
10
20
1
16
3
4
2
9
```