Coding Two: Advanced Frameworks
Assignment Element 1: Lab Work
Week 4 Exercise - Python webscraper
Build a simple webscraper that scrapes a set of documents from the internet and summarises them using gensim.
If you manage to achieve this, extract keywords from all the different documents and see if any are more popular than others.
Search for documents that contain those keywords using Python and then summarise those documents too.

Use BeautifulSoup to parse HTML

```
pip install beautifulsoup4
```

The import of the module is as follows:

```
from bs4 import BeautifulSoup
```

Let's take a look at the use of BeautifulSoup, we use the following HTML file to test:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <img class="test" src="1.jpg">
    <img class="test" src="2.jpg">
    <img class="test" src="3.jpg">
    <img class="test" src="4.jpg">
    <img class="test" src="5.jpg">
    <img class="test" src="6.jpg">
    <img class="test" src="7.jpg">
    <img class="test" src="8.jpg">
</body>
</html>
```

The above is a very simple html page, the body contains 8 img tags, now we need to get their src, the code is as follows:

```
from bs4 import BeautifulSoup


# Read html file
f = open('test.html', 'r')
str = f.read()
f.close()


# Create a BeautifulSoup object, the first parameter is the parsed string, and the second parameter is the parser
```

```
soup = BeautifulSoup(str, 'html.parser')


# Matching content, the first one is the tag name, the second one is the qualified attribute, the following shows the matching img tag whose class is testimg_list = soup.find_all('img', {'class':'test'})


# Iterate over tags
for img in img_list:
        # Get the src value of the img tag
        src = img['src']
        print(src)
```

The analysis results are as follows:

```
        1.jpg
        2.jpg
        3.jpg
        4.jpg
        5.jpg
        6.jpg
        7.jpg
        8.jpg
```

We can parse the web page and parse out the src in it, so that we can crawl image and other resource files. Below we use pear video as an example to crawl the video. The homepage URL is as follows: https://www.pearvideo.com/. We can see the following page by right-clicking and checking:
We can click 1 first, and then select the position that needs to be crawled, such as 2, and it will jump to the corresponding position on the right. We can see that there is an a tag on the outer layer. In our actual operation, we found that the webpage was redirected to the position where 2 was clicked. The webpage that we analyzed should be the herf value in the a tag. Because the herf value starts with /, the complete URL should be the main site + href value. Knowing this, we can proceed to the next step. Let's crawl the redirected url from the main site:

```
import requests
from bs4 import BeautifulSoup

# Main site
url = 'https://www.pearvideo.com/'
# Simulate browser accessheaders = {
        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 Safari/537.36'
}
# send request
response = requests.get(url, headers=headers)
```

```
# Get the BeautifulSoup object
soup = BeautifulSoup(response.text, 'html.parser')
# Parse out the a tag that meets the requirements
video_list = soup.find_all('a', {'class':'actwapslide-link'})
# Iterate over tags
for video in video_list:
    # Get herf and group it into a complete url
    video_url = video['href']
    video_url = url + video_url
    print(video_url)
```

The output is as follows:

```
https://www.pearvideo.com/video_1674906
https://www.pearvideo.com/video_1674921
https://www.pearvideo.com/video_1674905
https://www.pearvideo.com/video_1641829
https://www.pearvideo.com/video_1674822
```

We only need to crawl one. We went to the first URL to view the source code and found this sentence:

```
var contId="1674906",liveStatusUrl="liveStatus.jsp",liveSta="",playSta="1",autoPlay=!1,isLiving=!1,isVrVideo=!1,hdflvUrl="",sdflvUrl="",
hdUrl="",sdUrl="",ldUrl="",srcUrl="https://video.pearvideo.com/mp4/adshort/20200517/cont-1674906-15146856_adpkg-ad_hd.mp4",
vdoUrl=srcUrl,skinRes="//www.pearvideo.com/domain/skin",videoCDN="//video.pearvideo.com";
```

Among them, srcUrl contains the website of the video file, but we certainly can't find a webpage by ourselves. We can use regular expressions:

```
import re
# Get the source code of a single video web page
response = requests.get(video_url)
# Match video URL
results = re.findall('srcUrl="(.*?)"', response.text)
# Output
print(results)
```

The result is as follows

```
['https://video.pearvideo.com/mp4/adshort/20200516/cont-1674822-14379289-191950_adpkg-ad_hd.mp4'
```

Then we can download this video:

```
with open('result.mp4', 'wb') as f:
    f.write(requests.get(results[0], headers=headers).content)
```

The complete code is as follows:

```
    import re
    import requests
    from bs4 import BeautifulSoup
    # main site
```

```
url = 'https://www.pearvideo.com/'

# Simulate browser access

headers = {

        'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/67.0.3396.99 
Safari/537.36'

}

# send request

response = requests.get(url, headers=headers)

# Get the BeautifulSoup object

soup = BeautifulSoup(response.text, 'html.parser')

#Parse out the a tag that meets the requirements

video_list = soup.find_all('a', {'class':'actwapslide-link'})

# Iterate over tags

video_url = video_list[0]['href']



response = requests.get(video_url)



results = re.findall('srcUrl="(.*?)"', response.text)



with open('result.mp4', 'wb') as f:

    f.write(requests.get(results[0], headers=headers).content)
```