A spider to collect & preprocess the Data

CISC3025 - Natural Language Processing

Victor Mai Jiajun

March 21, 2025

1 Brif of project

The main objective of this project is: to write a spider using Python scripts to collect textral data at the given website **Books to Scrape** and preprocess them into a final corpus. The corpus is stored in the project folder as a `.json` file.

This is an individual project. I had done the 3 main functionalities in the project:

1. **Extract book URLs from a given website**: To cover the books from pages 1 to 10, we need to apply the usages of `requests` and `BeautifulSoup` to gather titles and urls corresponding to the 200 books.
2. **Preprocess on book description**: Based on the returned by `requests` elements, the description of each book are found and packed into a . `json` file, including the pre-processed source text of the title and the raw description.
3. **Preprocess descriptions and save them as relevant files**: By preprocessing the raw descriptions, tokenization, and other operations, the program generates one corpus corresponding to each book, which contains a list of words.

2 Extract book URLs from a given website(`spider_urls.py`)

2.1 Key methods

- Use `parse_pages()` to gather the book titles and their urls.
- In `parse_pages()`, apply `get_html()` and `parse_html()` to access web pages and parse their content while extracting the required headings, respectively.
- Use **codecs** to save the parsed content as a `.json` file.

2.2 Relevant codes

2.2.1 `parse_pages()`

```python
def parse_pages(start,end):
    #returns the book information on page between range(start,end).
    final_out = []
    for i in range(start,end):
        html_file = get_html('http://books.toscrape.com/catalogue/category/books_1/page-'+str(i)+'.html')
        temp_out = parse_html(html_file)
        final_out=final_out+temp_out
    return final_out
```

Listing 1: To parse 10 page of the certian website

2.2.2 get_html()

```python
def get_html(url):
    html_file = requests.get(url, headers={'User-Agent': 'Mozilla/5.0 \
                                (Macintosh; Intel Mac OS X 10_11_2) \
                                AppleWebKit/537.36 (KHTML, like Gecko) \
                                Chrome/47.0.2526.80 Safari/537.36'}).content
    time.sleep(random.random() + 3)  # break time
    return html_file
```

Listing 2: Request on html

2.2.3 parse_html()

```python
def parse_html(html_file):
    # returns the books tuple to acquire the url and book title
    soup = BeautifulSoup(html_file, "lxml")
    item_soup = soup.findAll(
        'article', attrs={'class': 'product_pod'})
    out = []
    try:
        for item in item_soup:
            title = item.h3.a['title']
            url = item.h3.a['href']
            out.append((title,url))
    except:
        logger.debug("Something goes wrong here.")
        traceback.print_exc()
        return None
    return out
```

Listing 3: To parse the content and get title

3 Preprocess on book description(spider_books.py)

3.1 Key methods

- Use get_book_description() to get the book descriptions from the content retrieved from urls.
- In get_book_description(), apply get_html() and parse_html() to access web pages and parse their content, respectively.
- In parse_html(), use a custom function idx_loop_contents() to decompose the content and find the description that I want.
- Finally, save the short title and raw texts of description into .json file.

3.2 Relevant codes

3.2.1 main()

```python
def main():
    """ Main Function """

    #get url from json file
    with open('url.json', 'r', encoding='utf-8') as f:
        url_list = json.load(f)
        f.close()
    #acquire book description from the url
    for item in tqdm(url_list):
        book_title = item[0]
        accessible_title = re.sub(r'[<>:"/\\|?*]', '-', book_title) # accessible in windows sys
        short_title = accessible_title[:50] # avoid long title

        book_description = get_book_description(item[1],base_url='http://books.toscrape.com/catalogue/')
        with open('test/'+short_title+'.txt','w', encoding='utf-8') as bookfile:
            bookfile.write(book_description)
            bookfile.close()

    logger.info("All Done!")
```

Listing 4: The main function of spider_books.py

3.2.2 get_book_description()

```
def get_book_description(url,base_url):
    book_url = base_url + url.strip('../../')
    return parse_html(get_html(book_url))
```

Listing 5: To get the description from the url given

3.2.3 get_html() & parse_html()

```
def get_html(url):
    html_file = requests.get(url, headers={'User-Agent': 'Mozilla/5.0 \
                                 (Macintosh; Intel Mac OS X 10_11_2) \
                                 AppleWebKit/537.36 (KHTML, like Gecko) \
                                 Chrome/47.0.2526.80 Safari/537.36'}).content
    time.sleep(random.random() + 3)  # break time
    return html_file

def parse_html(html_file):
    # get the book description
    soup = BeautifulSoup(html_file, "lxml")
    try:
        #DONE: Use the BeautifulSoup object to find the description of the book
        description = str(idx_loop_contents(soup)).encode('utf-8').decode('utf-8')
        return description
    except:
        logger.debug("Something goes wrong here.")
        traceback.print_exc()
        return None
```

Listing 6: To parse the content and get description

3.2.4 idx_loop_contents()

```
def idx_loop_contents(soup:BeautifulSoup, idx=None) -> NavigableString:
    # it is ok to find the description from F12 in chrome without using codecs
    if idx is None:
        idx = [5, 5, 3, 1, 5, 3, 1, 7, 0]
    cursor = soup
    for i in idx:
        cursor = cursor.contents[i]
    return cursor
```

Listing 7: To find the place of description accurately

4 Preprocess descriptions and save them as relevant files(`preprocess.py`)

4.1 Key methods

- Use `text_strip()`, `text_tokenize()`, and `text_stem()` to:
  - remove **"...more"** at the end of each text by regular expression.
  - remove abundent space of each text by regular expression.
  - lowercase letters.
  - tokenize text into token list by `word_tokenize` from `nltk.tokenize`.
  - stem the tokens by `PorterStemmer` from `nltk`.
- Save the pre-processed files into `.json` files.

4.2 Relevant codes

4.2.1 `main functions`

```python
def preprocess_corpus(corpus):
    for i in tqdm(range(len(corpus))):
        #strip some symbol from the original text
        stripped_text = text_strip(corpus[i][1])
        #use the tokenize function to tokenize the text
        tokenized_text = text_tokenize(stripped_text)
        #use the stemmer to stem the tokenized words
        for j in range(len(tokenized_text)):
            tokenized_text[j] = text_stem(tokenized_text[j])
        corpus[i][1] = tokenized_text
    return corpus
def text_strip(text):
    #DONE:Complete the function to finish the following task:
    text = re.sub(pattern=r'\.\.\.more$', repl='', string=text)
    text = re.sub(pattern=r'[\s]+',repl=' ',string=text)
    text = text.lower()
    return text
def text_tokenize(text) -> list:
    #DONE: Use the NLTK package or regular expression and string operations to
    tokenize the text
    tokens = word_tokenize(text)
    #The function should return a list that contains the tokenized words as the
     elements.
    return tokens
def text_stem(word:str) -> str:
    #DONE: Use the stemmer to stem the tokenized word
    #The function receives a word that needs to be stemmed and returns the
    stemmed word.
    stemmer = PorterStemmer()
    stemmed_word = stemmer.stem(word)
    return stemmed_word
```

Listing 8: The main functionality of preprocess.py

5 Conclusion

5.1 What i had learnt?

- When trying to collect data automately, we can make full use of python's `codecs` and `bs4` to spider the information and decode them to get the desired content.
- When pre-processing textral data, regular expressions are good enought for filtering illegal content and abundent spaces in sentences.

- When passing information between different python scripts, the data can be stored and used using the `.json` format.

5.2 To summarize:

For learning in the field of NLP, it is very important to master the skills of database and corpus construction.
We have to make full use of the advantages of computers, and by crawling information online and processing it offline, we can obtain quality training content suitable for machine learning in the future.I can finish this project easily, and I also encounterd with the unique charm and function of spider in python, which makes me more motivated for the study of NLP.