CS172 - Course Project
Search Engine: Web Scraper for Wikipedia

Kevin Nguyen
knguy523

1 INTRODUCTION

For this project, I decided to focus on building a search engine for the web. I collected documents from Wikipedia starting from the seed URL and followed any links on that page. I choose to start on a page about dogs; hopefully, the data reflects that.

2 TECHNICAL INFORMATION

Our web crawler uses Python 3.10 and the Scrapy framework[1]. To install the external libraries, a requirements.txt file is included. Below is a table that summarizes the external libraries used

Library	Usage
Scrapy==2.6.1	Web scraper framework
trafilatura==1.2.1	Main content text discovery and retrieval
pymongo==4.1.1	Interacting with MongoDB

2A Architecture

As mentioned above the web scraper uses the Scrapy framework. The framework abstracts most of the details allowing the developer to focus mainly on building the scraper and what to do with the data.

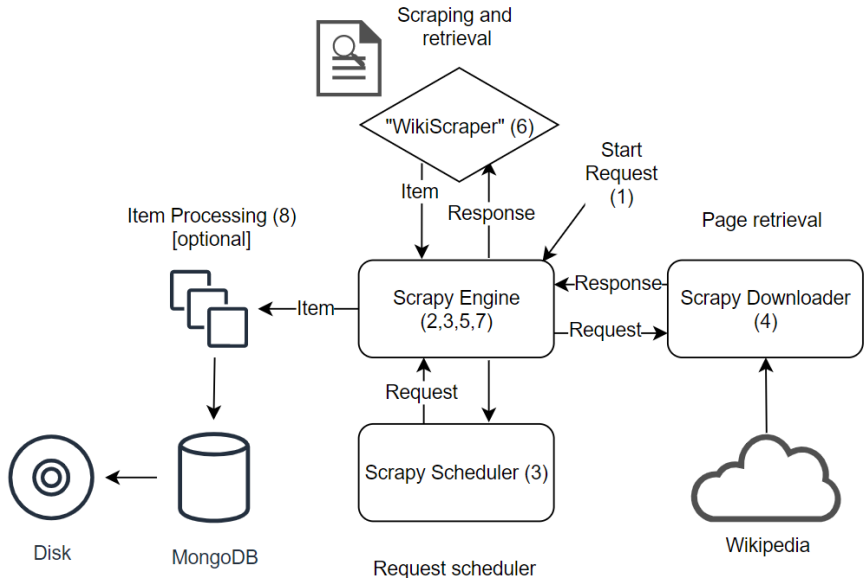

Figure 1: Data Path of Scrapy Spider

[1] https://scrapy.org/

Figure 1 provides a general flow of data in the system. The diagram was created by referencing the diagram explaining Scrapy architecture on the official docs[2]. Implementation of the spider and item processing scrape Wikipedia pages and send items to MongoDB if enabled. A functional local storage pipeline is implemented but I chose to use Scrapy's built-in “Feeds” exporter. When an output directory is specified when running the program, webpages will be stored individually as json files in the directory.

2B Data collection

A single crawler makes requests and processes the response from the Wikipedia pages. Information such as the url, title of the page, and main text are extracted from the HTML response to build the item.

Because of Wikipedia’s use of tables, text formatting, hyperlinks, lists and more, extracting the main body of the content proved to be a challenge. Just selecting paragraph elements would often grossly under select the content on the page. Using a more greedy css selector (e.g. grabbing all text from children of the main content tag) would often have undesired results like grabbing the names of other tags. Instead, Trafilatura[3] is used to extract the text from the page. Although not perfect, Trafilatura maintains the ordering of the text, which may be important for snippet generation.

Hyperlinks found within the main body are extracted and used to make more requests. Regex matching filters links to non-Wikipedia domains in addition to Wikipedia pages that do not have meaningful content. These include but are not limited to files, help & information pages, and discussion boards. Fortunately Wikipedia has reserved paths for these page called namespeaces[4]. Regex negative lookahead is used to filter out these specific namespaces.

```
def parse(self, response):
    log.info('Parse function called on %s on spider %s', response.url,self.name)
    #pass search to api to fetch main content
    content = extract(response.body,favor_recall=True,include_tables=False,include_comments=False,no_fallback=True)
    modRaw = response.css('#footer-info-lastmod').re_first('\d.*\d')
    lastMod = datetime.strptime(modRaw,"%d %B %Y, at %H:%M")
    # pattern to filter out wikipedia namespaces
    pattern = ('^\/wiki\/'
               '(?!(?:File:|Help:|User:|Talk:|Draft:|Media:|Module:|Portal:|Special:|Category:'
               '|Template:|MediaWiki:|TimedText:|Wikipedia:'
               '|User_talk:|Wikipedia_talk:|File_talk:|MediaWiki_talk:|Template_talk:|Help_talk:'
               '|Category_talk:|Portal_talk:|Draft_talk:|TimeText_talk:|Module_talk:)).*$')
    validLinks = set(response.css('#content * a::attr(href)').re(pattern))

    yield {
        'url': response.url,
        'title': response.css('title::text').get(),
        'lastMod': lastMod,
        'TOC': response.css('.toc .toctext::text').getall(),
        'mainText': content,
        'boldText': ' '.join(response.css('#content * b::text').re('.{2,}')),
        'italicText': ' '.join(response.css('#content * i::text').re('.{2,}')),
        'img': response.css('.infobox img::attr(src), .infobox-full-data img::attr(src), .thumbinner img::attr(src)').get(),
        'links': ' '.join(validLinks)
    }
```

Figure 2: Example Data Extraction the Crawler Uses

[2] https://docs.scrapy.org/en/latest/topics/architecture.html
[3] https://trafilatura.readthedocs.io/en/latest/
[4] https://en.wikipedia.org/wiki/Wikipedia:Namespace

Figure 2 showcases Scrapy's selector methods for data extraction. A dictionary of attributes is yielded from the spider to be processed by the item pipelines. Figure 3 is an example document stored in MongoDB.


```
  url: "https://en.wikipedia.org/wiki/Dog"
> TOC: Array
  boldText: "familiaris dog domestic dog dog"
  img: "//upload.wikimedia.org/wikipedia/commons/thumb/3/33/Blue_merle_koolie_…"
  italicText: "canis canis familiaris aegyptius alco americanus anglicus antarcticus …"
  lastMod: 2022-05-02T22:32:00.000+00:00
  links: "/wiki/Polar_bear /wiki/Spotted_hyena /wiki/Asian_golden_cat /wiki/Cur …"
  mainText: "dog the dog or domestic dog canis familiaris 4 5 or canis lupus famili…"
  title: "Dog - Wikipedia"
```


Figure 3: Example Document

Shown in Figure 4, I have collected about 102k documents totaling to about 1.35GB. However when exporting from MongoDB the file size reports just over 2.4GB

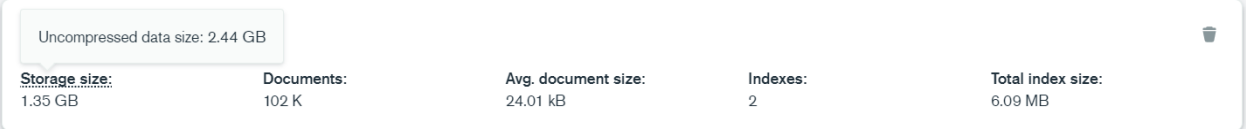

Figure 4: MongoDB Statistics

2C Data Structures

Behind the scenes, Scrapy uses a set data structure filled with url fingerprints for duplicate checking. Within the spider, I utilize sets to build a list of unique valid links to make a request. Apart from these, Scrapy's css selectors return python lists to which I converted into strings for text fields.

Scrapy's scheduler uses LIFO queues by default to perform DFO crawls[5]. BFO crawls can be performed by tweaking the setting to use a different queue data structure, but I left it as is.

2D Limitations

Scrapy is built upon Twisted which does not support multithreading. Additionally, source code utilizes singular threads. To try and improve concurrency, I tested configurations to run two spiders in the same script. However, this did not offer much improvement because it was simply alternating between spiders. It also relies on checking the presence of a URL among the documents. An index on the URL can solve this issue but synchronization between the crawlers is still an issue. Synchronization can be solved with a global set shared between crawlers, but that adds more memory overhead.

[5] https://docs.scrapy.org/en/latest/faq.html

If more processing is required on the collected data, perhaps I would be able to take advantage of multiple crawlers with more middleware and pipelines. For my use case, defining multiple spiders crawling the same domain did not make much sense.

Despite this limitation of the Twisted, Scrapy is built for concurrency and asynchronous events. The setting `CONCURRENT_REQUESTS` and configuration of AutoThrottle enable Scarpy to optimize throughput while still being friendly to the website. As a result, Scrapy is still very fast[6].

The performance of the web scraper, therefore, boils down to information collected with the selectors. The use of Trafilatura to extract the main text was a trade-off between accuracy and speed. Wikipedia does have an API that can conveniently send the raw text of an article. However, this requires an additional set of requests and responses, which would dramatically reduce performance. Using the API would also put additional strain on Wikipedia and its usage rules.

Another limitation of the system is that it lacks generalizability. Apart from Trafilatura, which is built to extract the main content, the CSS selectors employed are site-specific. Fortunately, the structure between pages is consistent. Retrieving the image URL is trickier since they reside in different parts of the page, but this is overcome by searching typical locations and grabbing the first element.

Apart from those limitations, there are some minor quirks. As of now, there is little to no processing on the text retrieved from the webpage. Although not necessarily a limitation, it may affect the processing in Part B. Just in case, I have stored a separate MongoDB collection with lowercasing, punctuation removed, and extraction of only the ASCII alphabet.

Due to Scarpy abstraction and handling of errors, non-critical errors may cause the program to stall. Such an example is when there are connection errors with MongoDB to instantiate the client in the pipeline. There are options to close the spider if this happens, however, I was unable to figure out how to send a signal to the Twisted reactor to terminate the program after closing the spider. To work around this issue, I made the MongoDB connection optional, which also improves performance when only writing documents to the local file system.

3 INSTRUCTIONS TO DEPLOY

The web scraper is defined in a Scrapy project. You will need to install the required libraries with the provided requirement.txt. It is recommended that the libraries are installed in a python virtual environment. An example command using pip: `pip install -r requirements.txt`

If needed, start an instance of MongoDB and provide connection credentials at the bottom of the wikipedia_scraper/settings.py. By default with MongoDB enabled, it will try to connect to a

[6] https://docs.scrapy.org/en/latest/topics/benchmarking.html

localhost instance of ‘mongodb://localhost:27017’ with database name ‘Wikipedia’ and collection name ‘Pages’.

After installing the requirements, a bash script is provided to run the program. It simply runs the python script with some arguments: `./run.sh <seed.txt> <page_limit> <hop-away> <output-dir>`. The seed file provided must contain the URLs delimited by a new line.

The script can also be run using `python <path-to-wikiscrape.py>`. Use the flag -h to print out the usage options.

4 SAMPLE OUTPUT

```
(venv) C:\Users\Kevin\Documents\GitHub\WikipediaScraper>C:\Users\Kevin\Documents\GitHub\WikipediaScraper\wikipedia_scraper\spiders\wikiscrape.
py -h
usage: wikiscrape.py [-h] [-p #] [-d #] [-o dir] [-m] [-s] [input_file]

Crawler options. To configure mongoDB connection, please edit project settings.py

positional arguments:
  input_file            Input seed file of urls

options:
  -h, --help            show this help message and exit
  -p #, --num_page #    Page limit on crawl (default = inf)
  -d #, --depth #       Depth limit on crawl (default = inf)
  -o dir, --output dir  Output directory to store scraped pages. Creates directory if doesn't exist
  -m, --mongo           Option to enable mongo pipeline. Configure settings in settings.py
  -s, --silent          Option to supress logging to INFO
```

Figure 5: Program usage

```
$PWD\wikipedia_scraper\spiders\wikiscrape.py seed.txt -p 100 -d 1 -o pages -s
```

```
2022-05-08 12:33:34 [scrapy.core.engine] INFO: Spider opened
2022-05-08 12:33:34 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2022-05-08 12:33:34 [scrapy.extensions.telnet] INFO: Telnet console listening on 127.0.0.1:6023
2022-05-08 12:33:41 [__main__] INFO: Parse function called on https://en.wikipedia.org/wiki/List_of_dog_breeds on spider wiki
2022-05-08 12:33:41 [trafilatura.core] INFO: not enough comments None
2022-05-08 12:33:42 [scrapy.extensions.feedexport] INFO: Stored json feed (1 items) in: pages/1-page2022-05-08T19-33-34.885581.json
2022-05-08 12:33:44 [__main__] INFO: Parse function called on https://en.wikipedia.org/wiki/List_of_horse_breeds on spider wiki
2022-05-08 12:33:44 [trafilatura.core] INFO: not enough comments None
2022-05-08 12:33:44 [scrapy.extensions.feedexport] INFO: Stored json feed (1 items) in: pages/2-page2022-05-08T19-33-42.330744.json
2022-05-08 12:33:44 [__main__] INFO: Parse function called on https://en.wikipedia.org/wiki/Laika_(dog_type) on spider wiki
2022-05-08 12:33:45 [trafilatura.core] INFO: not enough comments None
2022-05-08 12:33:45 [scrapy.extensions.feedexport] INFO: Stored json feed (1 items) in: pages/3-page2022-05-08T19-33-44.415448.json
2022-05-08 12:33:46 [__main__] INFO: Parse function called on https://en.wikipedia.org/wiki/Grand_Anglo-Fran%C3%A7ais_Blanc_et_Orange on spider wiki
2022-05-08 12:33:46 [trafilatura.core] INFO: not enough comments None
2022-05-08 12:33:46 [scrapy.extensions.feedexport] INFO: Stored json feed (1 items) in: pages/4-page2022-05-08T19-33-45.088167.json
```

Figure 6: Sample Output with Logging Level = INFO