Parsing Tables from EDGAR with Python and Its Dockerization

Jiali Cheng[1]
[1]Graduate School of Engineering, Northeastern University, Boston, USA

***Abstract*—We use Python to access data from EDGAR, the Electronic Data Gathering, Analysis, and Retrieval system. Given a company ID, called CIK, and an Access Number, called acc-no, an URL to this HTML file is generated. We locate to the 10Q file and extract all the tables and save them as CSV(Comma Separated Values) files. Then we dockerize this pipeline so that it can be used for any websites, not only restricted to IBM.**

***Keywords*—Parse Tabels, HTML, EDGAR, Python, Docker**

I. Introduction

The Electronic Data Gathering Analysis and Retrieval (EDGAR) system is run by Office of Information Technology, U.S. Securities and Exchange Commission. performs automated collection, validation, indexing, acceptance, and forwarding of submissions by companies and others who are required by law to file forms with the SEC. The database is freely available to the public via the Internet (Web or FTP).

In this paper we focus on parsing the tables in the 10Q file of each company. And we introduces a way that given a CIK (company ID), automatically generates the URL (Uniformly Resource Location) to the 10Q file and parse all the tables to save to a CSV file. We then build this pipeline into a Docker image so that it can run on other PCs or laptops.

Parsing tables have always been a hot topic and a useful tool for data collection. We use

The remainder of the paper is organized as follows. In Sect. II we briefly go through the objectives and expectations of the TableParser package and provide an overview of it. Then we detail the design of this package in Sect. III. Sect. **??** presents the results and analysis of the algorithm and code. And in Sect. **??** we conclude the report.

II. System Model

According to the issues addressed in Sect. II, we develop a package called TableParser. It basically follows the processing order shown in Fig. 1. Given an Accession Number, we can easily generate the URL of its files on the EDGAR system. And we use regular expression to locate to the 10Q file of this company, which is the target of our parsing job. Then we call the read_html function to parse the tables out. We clean the noisy data and write it to a CSV file.

III. The Package

In this section we describe the detailed design of each part of the package. The package has 2 modules, preparation and parsing. The preparation module basically gets the Accession Number, generates the URL and locates to the 10Q file. And then the parsing module is in charge of extracting the tables and save the items into a CSV file.

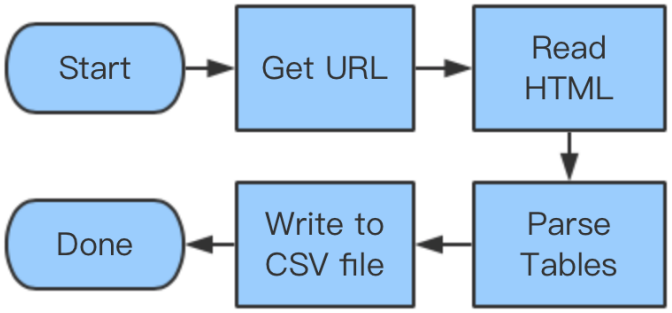

Fig. 1. The processing flowchart

First we generate URL according to the given acc_no. The logic is shown below:

Algorithm 1 Get URL

Input: acc_no
Output:
1: **function** GET URL(string acc_no)
2: GETCIK(acc_no)
3: CLEANDASH(acc_no)
4: GENERATEURL(acc_no)
5: LOCATE_TO_10Q(url) ▷ Using regular expresion
6: **end function**

Then we delve into the parsing module. As shown in Algorithm III, the function $extract()$ takes in the URL of the 10Q file generated by its former part and get the HTTP response. Then it transforms the response into lxml string form for later use. Then we use the package BeautifulSoup to parse all the tables. We utilize the function find_all that can find all the tags in an HTML string. It is not difficult to draw the basic idea as follow:

1) Find all the table tags
2) Within each table, find all the 'th' tags, which stands for rows
3) Within each row, find all the 'td' tags, which stands for columns
4) Write each column item into the CSV file

which is shown in Algorithm III.

When an Accession Number is given, the program initiates a unique instance that is related to the Accession Number. If you want to start the parsing process, hit Y and the program will call the the function $start()$, which consists of prepare for the URL of the 10Q file and start extracting. The pseudocode is shown in Algorithm III. The result of the parsed tables from the given URL is as follow:

Algorithm 2 Extract, extracting tables from the 10Q file

Input: url_10q
Output: double k ▷

Get an HTTP response according to the URL of 10Q file and transform the response into lxml form. Then find the HTML tags to get the table items.

```
1: function MASTER(double curTime)
2:     response ← RESPONSE(url_10q)
3:     soup ← BEAUTIFULSOUP(response)
4:     filename ← acc − no
5:     OPEN_CSV_FILE()
6:     for doFIND_ALL('table')
7:         for doFIND_ALL('th')
8:             for doFIND_ALL('td')
9:                 WRITE_RESULT(filename)
10:            end for
11:        end for
12:    end for
13: end function
```

Algorithm 3 Start, start running the whole process

Output: double k

```
1: function MASTER(double curTime)
2:     url ← FORMURL(acc − no)
3:     url_10q ← FORMURL(acc − no)
4:     EXTRACT
5: end function
```

IV. Conclusion

In this paper, we propose a way to parse tables in the 10Q file of a company in the EDGAR system. Then we dockerize this program to use it on other platforms. Results show that we have finish the job and reach the objectives.

References

[1] Liebrecht, P., Schier, J., Bhasin, K., Bibyk, I., Butler, M., Hudiburg, J., Tai, W., Shames, P., NASA's Space Communications Integrated Architecture, Proceedings of SpaceOps 2010 Conference, AIAA, 25-30 April 2010, Huntsville, Alabama.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	
2	0																
3	1					Three Months Ended	S	Nine Months Ended		September 30,							
4	2	(Dollars in millions)			2013			2012			2013			2012			
5	3	Net income		$	4041		$	3824		$	10299		$	10771			
6	4	Other comprehensive income/(loss), before tax:															
7	5		Foreign currency translation ad		382				501			-959			164		
8	6		Net changes related to available-for-sale securities:														
9	7			Unrealized gains/(losses) arisi		3				11			0			13	
10	8			Reclassification of (gains)/los		-5				-27			-5			-43	
11	9			Subsequent changes in previously impaired													
12	10			securities arising during the pe		1				-7			3			20	
13	11		Total net changes related to av		-1				-24			-1			-10		
14	12		Unrealized gains/(losses) on cash flow hedges:														
15	13			Unrealized gains/(losses) arisi		-409				-54			-58			65	
16	14			Reclassification of (gains)/los		-27				-112			-130			-246	
17	15		Total unrealized gains/(losses)		-436				-165			-188			-181		
18	16		Retirement-related benefit plans:														
19	17			Prior service costs/(credits)		0				0			33			0	
20	18			Net (losses)/gains arising duri		105				1			300			66	
21	19			Curtailments and settlements		0				-2			0			-1	
22	20			Amortization of prior service (		-28				-37			-86			-112	
23	21			Amortization of net (gains)/los		872				613			2623			1846	
24	22		Total retirement-related benefi		949				575			2869			1799		
25	23	Other comprehensive income/(los			895			887			1721			1771			
26	24	Income tax (expense)/benefit related to items of															
27	25			other comprehensive income		-91				-109			-933			-606	
28	26	Other comprehensive income/(los			804			778			788			1165			
29	27	Total comprehensive		$	4844		$	4601		$	11087		$	11936			
30	28																
31	29	(Amounts may not add due to rounding.)															
32	30																
33	31	(The accompanying notes are an integral part of the financial statements.)															

Fig. 2. The flow diagram of the LCPR algorithm