Missing Data Analysis for the EDGAR Log File

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

***Abstract*—We use Python to access the EDGAR Log File Data Set and build a pipeline that dose the following. Given a year, the pipeline get data form the first day of the month for every month in the year. And we replace observable anomalies and handle missing data by filling with the mean value. Then we compute summary metrics, plot the results and present an analysis. The program logs all operations with a time stamp, compile 12 summaries into one file and upload it to the AWS S3 bucket.**

***Keywords*—Missing Data Analysis, EDGAR, Summary Metrics, Python, ASW S3 Bucket**

I. Introduction

The EDGAR Log File Data Set contains information in CSV format extracted from Apache log files that record and store user access statistics for the SEC.gov website. Due to certain limitations, including the existence of lost or damaged files, the information assembled by DERA may not capture all SEC.gov website traffic. Given the large size of the data files which can include more than a million entries, for best results users should avoid using software that limits the amount of data that can be read.

While in real life data processing, there dose exist some missing and anomaly values which we can not bypath. By ”missing” we simply mean null or ”not present for whatever reason”. Many data sets simply arrive with missing data, either because it exists and was not collected or it never existed. For example, in a collection of financial time series, some of the time series might start on different dates. Thus, values prior to the start date would generally be marked as missing.

And there also exist anomalies in the data sets. In data processing, anomaly means items, events or observations which do not conform to an expected pattern or other items in a dataset. Anomaly detection (also outlier detection) is the identification of these things. Typically the anomalous items will translate to some kind of problem such as bank fraud, a structural defect, medical problems or errors in a text. Anomalies are also referred to as outliers, novelties, noise, deviations and exceptions.

In this report, we target at analysing the EDGAR log file in presence of missing and potentially wrong data item. Facing different field of data and different types of missing and anomalies, we adopt accordingly to find and correct the anomalies, as well as handling missing data. We retrieve log files from the EDGAR system, find anomalies, deal with missing data, compute metrics and upload useful information to AWS S3 bucket in Python3. Then we build a package to reuse this whole functionality and dockerize this pipeline to work in other environment.

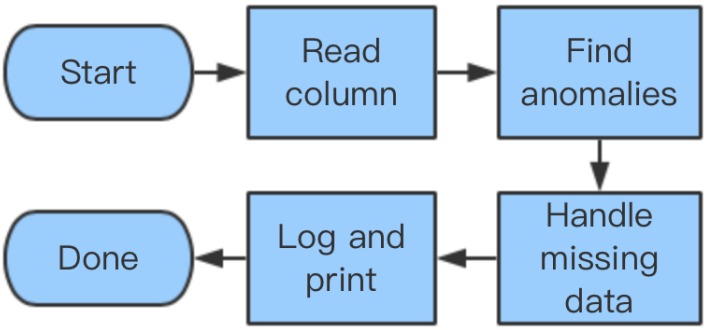

Fig. 1. The processing flowchar

The remainder of the paper is organized as follows. In Sect. II we talked about what the EDGAR log file has tracked and what’s our method according to each field of information. Then in Sect. III we provide the overview of the MissingData package that put our plan into action. Sect. IV presents the results and analysis of the algorithm and code and conclude the report.

II. System Model

We develop a package called MissingData to precess data from log files of the EDGAR system. It basically dose the following: handling missing data, compute summary metrics, finding observable anomalies, log all the operations, compile the files of a year and upload to a AWS S3 bucket.

The EDGAR log file captures 13 features of a request and we focus on the following: IP address, file size, whether using crawler and which browser. For each feature we handle the missing parts, find anomalies and compute specific metrics. Since there are both missing data and observable anomalies, we first deal with the anomalies and then fill the missing data with the mean value so that the anomalies would have less influence on the missing data, as Fig. 1 shows. The following are the precessing of each feature:

1) For IP address, which is a categorical data, we figure out which type of IP a request is from. This is not easy to fill the missing and potentially wrong ones since we hardly have clues in the log file to rely on. So we can nothing towards them.

2) For time, which is a numerical data, we correct errors like illegitimate time values. We fill the missing time value with the previous time, which is the easiest way to deal with

Fig. 2. The flow diagram of the LCPR algorithm

the missing parts, for the log of the operations are in time order. Then we plot the distribution of requests with respect to time.

3) For size, which is a numerical data, it is less possible to have extremely values, since people are accessing almost same type of files in regular size. So we replace anomalies and missing data with mean values.

4) Apache HTTP code, which is a categorical data, represents the type of the response. We use the mode to fill the blank and then plot the distribution of different kinds of response because we are most likely to get similar responses.

5) For using agent and crawler, which are categorical datas, we also use the mode to fill the missing data and plot the distribution of population using these techniques versus not.

6) For the browser field, which is a categorical data, we use the mode the fill the missing data. Then we plot the distribution of numbers of various browsers and get to the most-used browser.

III. The package

Following the precessing idea in Sect. II, we develop this MissingData package in Python3 to finish the jobs stated above. The package takes into the year that is to be processed and automatically get the log files of first day of each month. After analysing, it upload all the data into an AWS S3 bucket.

The package can be divided into two modules, the preparation and the analysing module. The former dose log file URL generating according to the year provided. While the latter dose all the analysis work. Hear we focus on the analysing module and detail some implementations of the pipeline we build.

IV. Result

The results of the concatenated data, metrics and logs are as follow: The final data, summary metric and log files are shown in Fig. 2. In this paper, we analysed the EDGAR log files, in presence of missing and potentially wrong data item. Facing different field of data and different types of missing and anomalies, we adopt accordingly to find and

	A	B	C	D	E
1		0	1	crawler	browser
2	class_a	294581			
3	class_b	145145			
4	class_c	0			
5	missing	452	0		
6	not using agent		157225		
7	total	440178	440178		
8	using agent		282953		
9	0			421584	
10	1			18594	
11	win				216624
12	mie				11324
13	mac				237
14	lin				140
15	opr				13
16	iem				1
17	class_a	99429			
18	class_b	92901			
19	class_c	0			
20	missing	1472	0		
21	not using agent		2810		
22	total	193802	193802		
23	using agent		190992		
24	0			182638	
25	1			11164	
26	win				106682
27	mie				14731
28	mac				456
29	lin				258
30	opr				38
31	iem				1
32	class_a	324950			
33	class_b	531266			

2004-final-summary

Fig. 3. The flow diagram of the LCPR algorithm

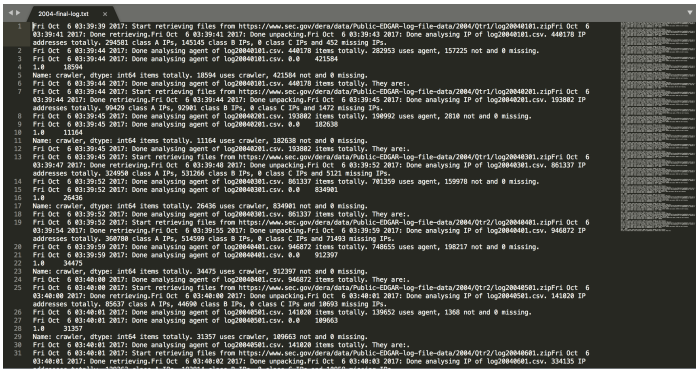

Fig. 4. The flow diagram of the LCPR algorithm

correct the anomalies, as well as handling missing data. We retrieve log files from the EDGAR system, find anomalies, deal with missing data, compute metrics and upload useful information to AWS S3 bucket in Python3. Then we build a package to reuse this whole functionality and dockerize this pipeline to work in other environment.

REFERENCES

[1] ttps://pandas.pydata.org/pandas-docs/stable/index.html

[2] Chandola, V.; Banerjee, A.; Kumar, V. (2009). "Anomaly detection: A survey". ACM Computing Surveys. 41 (3): 1?58. doi:10.1145/1541880.1541882