Technical report

AXEL LASSEUR, University of Piraeus, GREECE

ACM Reference Format:
Axel Lasseur. 2018. Technical report. 1, 1 (May 2018), ?? pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 GENESIS

The set of data is big (2.2G) and it's an XML file, a format which is more heavy to read than to store (because of the parsing process). Well, I tried to open it with a regular text edition tool and, as expected, my computer runned out of memory just trying to read the raw document. Further more, this format is higly deprecated and online help communities mostly suggest to use another format source... The obvious truth was finally set in my mind : this one is going to be tricky.

First of all, I checked how the file looks like by hand with the help of tail and head UNIX commands and I figured out that everything is wrapped into a '<dblp>' tag, and each child seems to be a publication with an mdate parameter. Among those children, I've seen the tags '<article>', '<mastersthesis>' and '<phdthesis>'. Here is the first lines of data :

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE dblp SYSTEM "dblp.dtd">
<dblp>
<article mdate="2017-05-28" key="journals/acta/Saxena96">
<author>Sanjeev Saxena</author>
<title>Parallel Integer Sorting and Simulation Amongst CRCW Models.</title
<pages>607-619</pages>
<year>1996</year>
<volume>33</volume>
<journal>Acta Inf.</journal>
<number>7</number>
<url>db/journals/acta/acta33.html</url>
<ee>https://doi.org/10.1007/BF03036466</ee>
</article><article mdate="2017-05-28" key="journals/acta/Simon83">
<author>Hans Ulrich Simon</author>
<title>Pattern Matching in Trees and Nets.</title>
<pages>227-248</pages>
<year>1983</year>
<volume>20</volume>
<journal>Acta Inf.</journal>
```

Author's address: Axel Lasseur, University of Piraeus, Piraeus, GREECE, axel.lasseur@etu.univ-lyon1.fr.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2018 Association for Computing Machinery.

XXXX-XXXX/2018/5-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

```
<url>db/journals/acta/acta20.html</url>
<ee>https://doi.org/10.1007/BF01257084</ee>
</article><article ...
...
...
```

After a few investigation, I decided to use javascript (nodejs) with big-xml library (https://www.npmjs.com/pa xml). This library search a tag according to a regex request and release the memory once one is found, then keep going the research. Ok, then let's try it with this one : `/^(article|mastersthesis|phdthesis)$/`

As a result, I get this dataset :

```
[ { year: 1936, n: 12 },
  { year: 1937, n: 15 },
  { year: 1938, n: 11 },
  { year: 1939, n: 18 },
  { year: 1940, n: 10 },
  ...
  { year: 2015, n: 127607 },
  { year: 2016, n: 137248 },
  { year: 2017, n: 149683 },
  { year: 2018, n: 51530 },
  { year: undefined, n: 3 } ]
```

Cool, this looks nice for a first shot. Well, my instinct told me at this moment that it seems to be too easy, and there are not so much data to play with at final. There is probably more than 3 children tags (article, masterthesis, phdthesis) and I need to list it in order to feed my script through the regular expression.

2 HACK THE DATA FILE

My javascript works well to match some children of `<dblp>` so, let's have a try on `<dblp>` itself to figure out the name of each child only. I write a new code under the name of `tagsname_getter.js`, but I know something : big-xml library search for a tag, then read the whole content and give the fallback before releasing the memory and going to the next tag whitch match with the regular expression. Yet, `<dblp>` is the only one tag which wrap the entire file. I don't like to be pessimistic but, I think it won't make it. After 5 minutes waiting for the program to give me an output, the smashing conclusion clearly appeared :

```
FATAL ERROR: CALL_AND_RETRY_LAST Allocation failed - JavaScript heap out of memory
```

I listed several ways to deal with this file :

- I split it in several files and I wrap each one with <dblp>, but it's very inaccurate and I'll probably spend lot of time on technical details
- I find something to put these data in a relational database, but it needs obviously an XML parser and I'll probably cope with the same memory issue
- I rent a powerful server to run my script, but it requires money and OS setup (at least to install node, ok it's easy but you never know!)
- I pray god but it only works in case of extrem emergency
- I grep it, it's a line by line reader, with regex possibility and without worry about memory managment x)

Without any surprise, I decided to pray god and grep it. I suddenly realized that I could just grep the whole document since the beginning but I already spent a good amount of time on my javascript and the object output is a good way to extend the program if required in the future (perhaps to get a dataset according to the type of publication, who knows ?)

This idea gave birth of this bash script below :

```
#!/bin/bash

DATA="dblp.xml"

echo "cutting␣the␣file␣$DATA..."

grep -o '<[a-z]\+ mdate' $DATA | sed -e 's/<\([a-z]*\) mdate/\1/' > tagsn

echo "tagsname.txt␣created.␣A␣list␣of␣unique␣tags␣name␣is␣processing,␣plea

TAGS=()

#read line by line
while IFS='' read -r line || [[ -n "$line" ]]; do
        #search in the TAGS array if the tagname is already recorded
        RECORDED=false
        for i in "${TAGS[@]}"; do
                if [ $i == $line ]; then
                        #echo yes
                        RECORDED=true
                        break
                fi
        done
        if [ $RECORDED = false ]; then
                echo $line
                TAGS+=($line)
        fi
        #echo "Text read from file: $line"
done < tagsname.txt

echo "Done␣:)"
```

with the following output :

```
article
proceedings
inproceedings
incollection
book
phdthesis
```

mastersthesis

www

:D

The final dataset can be found under the name of record.json

3 LET'S PLAY WITH THE DATASET

To show the data, I decided to use web languages with echarts, a JavaScript api to make charts. The data analysis is inside the charts directory

The techniques applied to the data are the moving average and the exponential moving average, all the code is in charts/index.html You are free to change the settings with the following const :

```
const MA_n = 10; //previous n data used for Moving Average
const EMA_n = 10; //previous n data used for Exponential Moving Average
const EMA_alpha = 0.5; //Îś value for Exponential Moving Average
```