Web Data Model Project Report

Author: **Mert Ozer**

Date: **17.11.2017**

Table of Contents

How to run the program?

```
python ./ dtdParser <pathToXml> <pathToDtd>
Example usage: python ./dtdParser.py input_file dtd_file
```

Python 3 is used in this project.

Implementation analysis

I did not use any external libraries in my code except for the creating huge xml's and plotting my results and sys library to give arguments from command line.

My data types:

- **node**: nodes have children, parent and name.
- **tree**: consist of nodes. Has a root element. We can add nodes, and get childs of a given element.
- **State**: state has a name, and can be linked to other states.
- **automata**: Consist of states. can add new states

So the basic idea behind my code is:

1 Create a tree from the xml file. If there is an error in the xml file stop the execution (I will talk later on how I am doing this)

2 For every rule in Dtd all the children of that element from the tree.

3 Merge the names of those children's and get a string let's name it str.

4 Create the automaton from Dtd file (I will talk later on how I am doing this)

5 repeat step 2-3-4 until end of the dtd file.

6 If there are no errors that means we have a matching dtd file and xml file. If there are some errors print them and stop the code execution.

Creating a tree from the xml file.

This is really easy to do with stack. If it is a new element add to stack, if not check pop and check if the element matches new element. If there is an error that means it is not a well formed xml. If all things go right, this means that we have a well formed xml. In this part, I believe my algorithm is optimal. I cannot see any other way to improve it.

Creating the automaton from dtd file.

For creating an automaton I did not follow any algorithms like Thompson or Glushkov. I created my own algorithm because they seemed not easy to implement. But honestly after looking to my automatan creation I can say that it is not easy to read my code also I think that we may not need to create automata and state classes. They are taking a lot of space in the memory. We can replace them by Glushkov algorithm and with that we will just have the table and from there we can do whatever we want. That is why I believe it could be improved. Also in automaton and states there are some unnecessary variables that I use in those classes. I will share the plots below, but when I replace my automaton with regex match function I saw really huge improvement in time wise that means there are still optimization left in my code.

After evaluating my code on large files, I saw that putting everything into memory could be a problem. If I knew it was going to be only large files I would have choose SAX method instead of putting them into a tree.

Experimental Evaluation

I created 2gb of xml files from my dtd file (below) with an external library. I tried to cover as much as different types of interaction between states. I believe it is a complex dtd.
I tried to make really complicated dtd files, but the size of the xml files are affecting the execution time more than dtd file size. So after looking my results, I can say that there is a correlation between my algorithm and regex library, but regex library is is way faster.

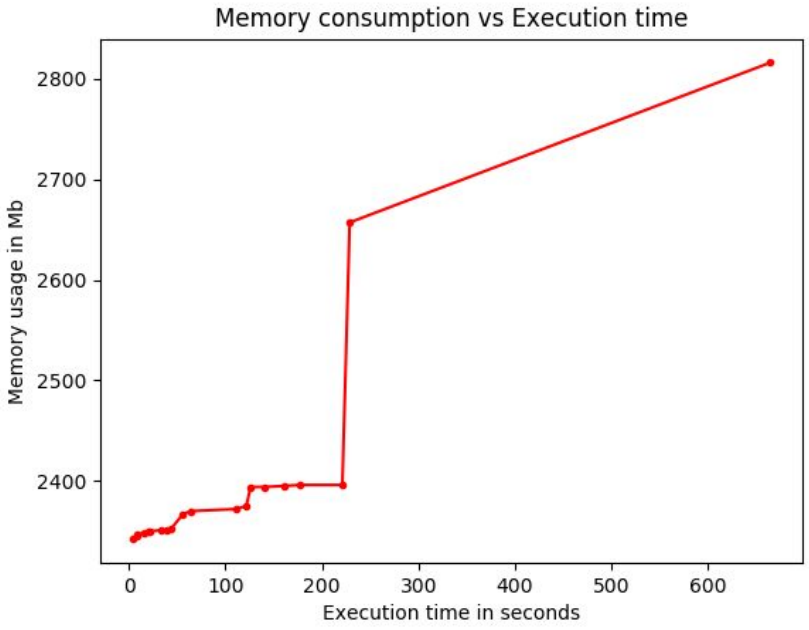
Here is the dtd file:

a b+c+d+n+h+(((f+)))v+x+m*o*e+g+i+
b kl+i*e(wt)+b?
c k+l+f+h+n+
d l*o+
k o+p*
l dv+
i l+e
e k+l+i?
w e+th+
t k+w?
o pq+
p o?m+
q u(x)+
u q?
m (qz)*(yr)*
z m?
x r*
r x?
v y+
y v?
f gl
g f?
h j
j h?
n ((s))
s n?

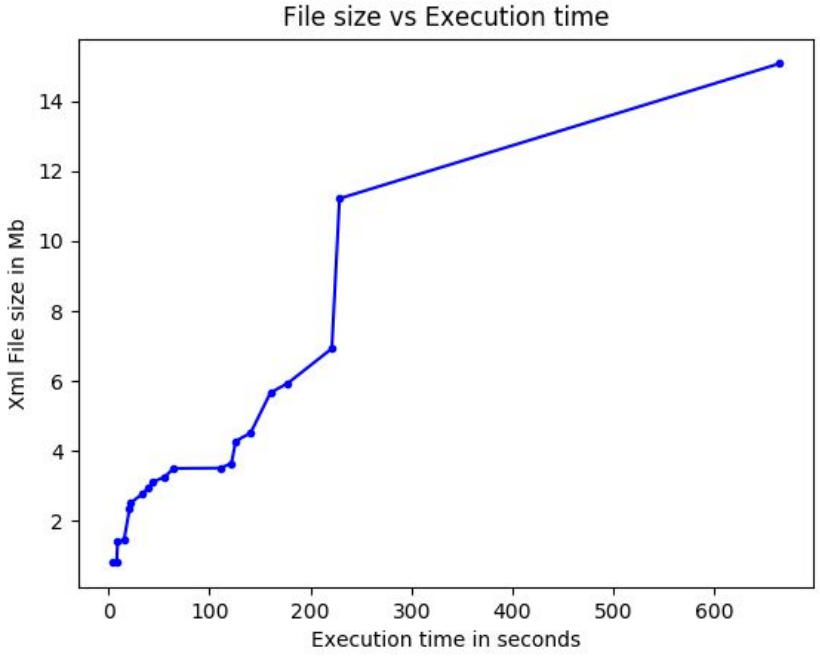
I did not try empty elements here because I am using an external library to create xml files according to my dtd file. And in my code I handle empty elements in a separate case. (Since I cannot create an automata for it)

Results for my algorithm

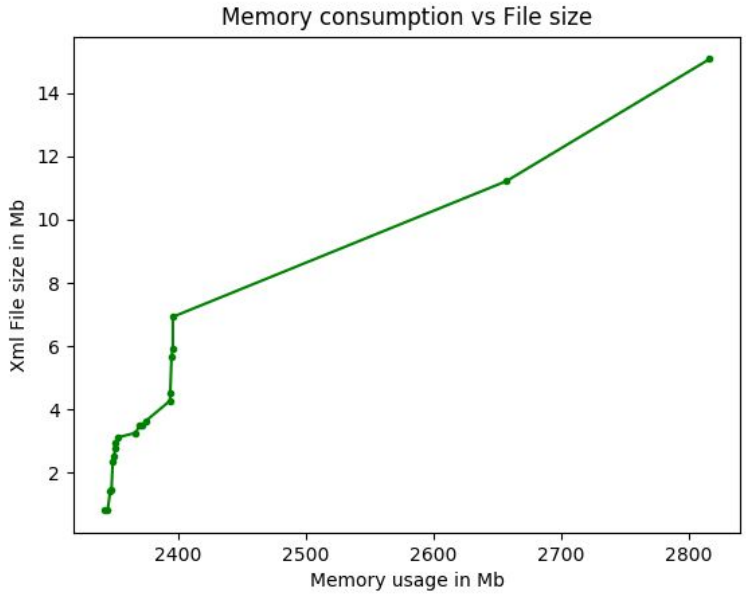
Memory consumption vs execution time



File size vs execution time

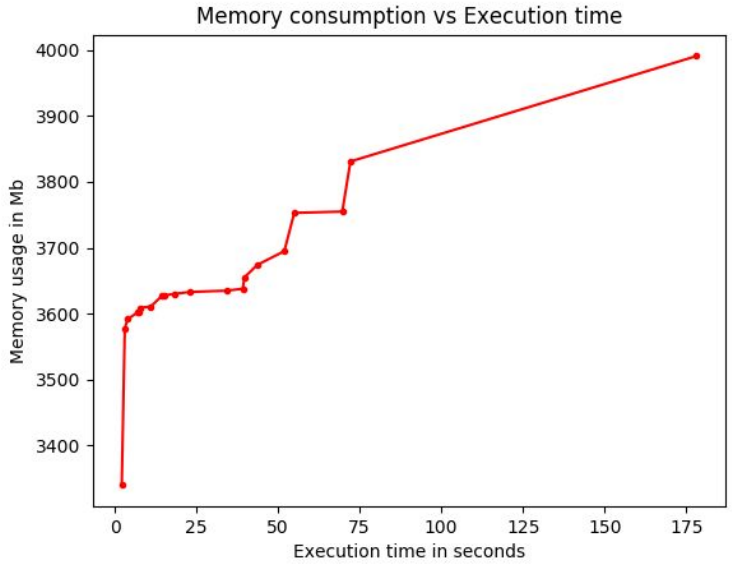


Memory consumption vs file size

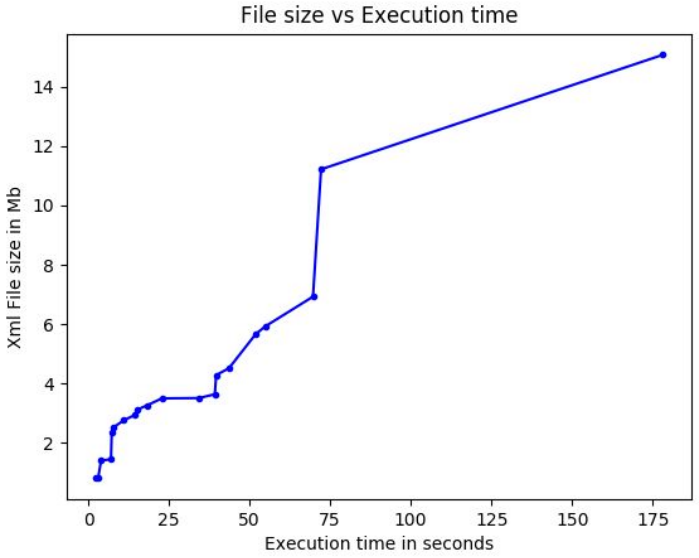


Results from regex matching

Memory consumption vs execution time



File size vs execution time



Memory consumption vs file size