Program & Profiling Report

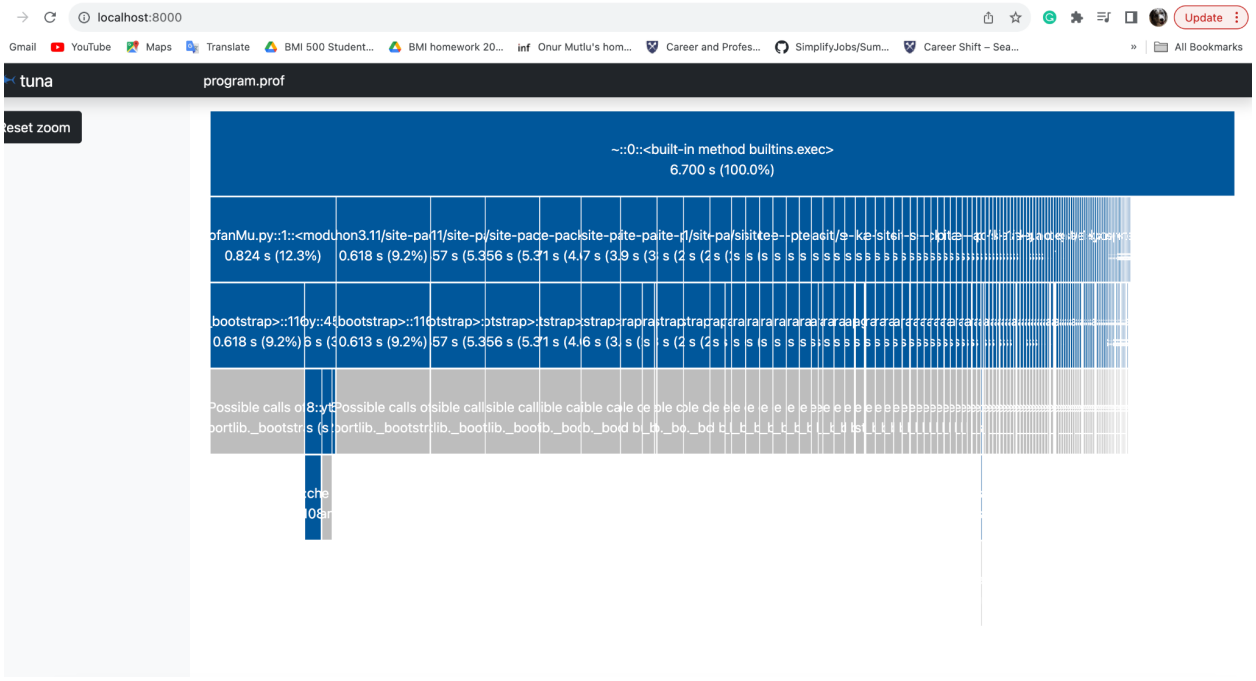
After examining the original code provided, I figured out that even though we are trying to check for different stuff, the logic of the deid_(blah) and check_for_(blah) should always remain the same. The only thing we need to modify to let our program check for different patterns should just be the regular expression pattern(originally called phone_pattern). For this program, I decided to try to check for the date, and here is my modification:

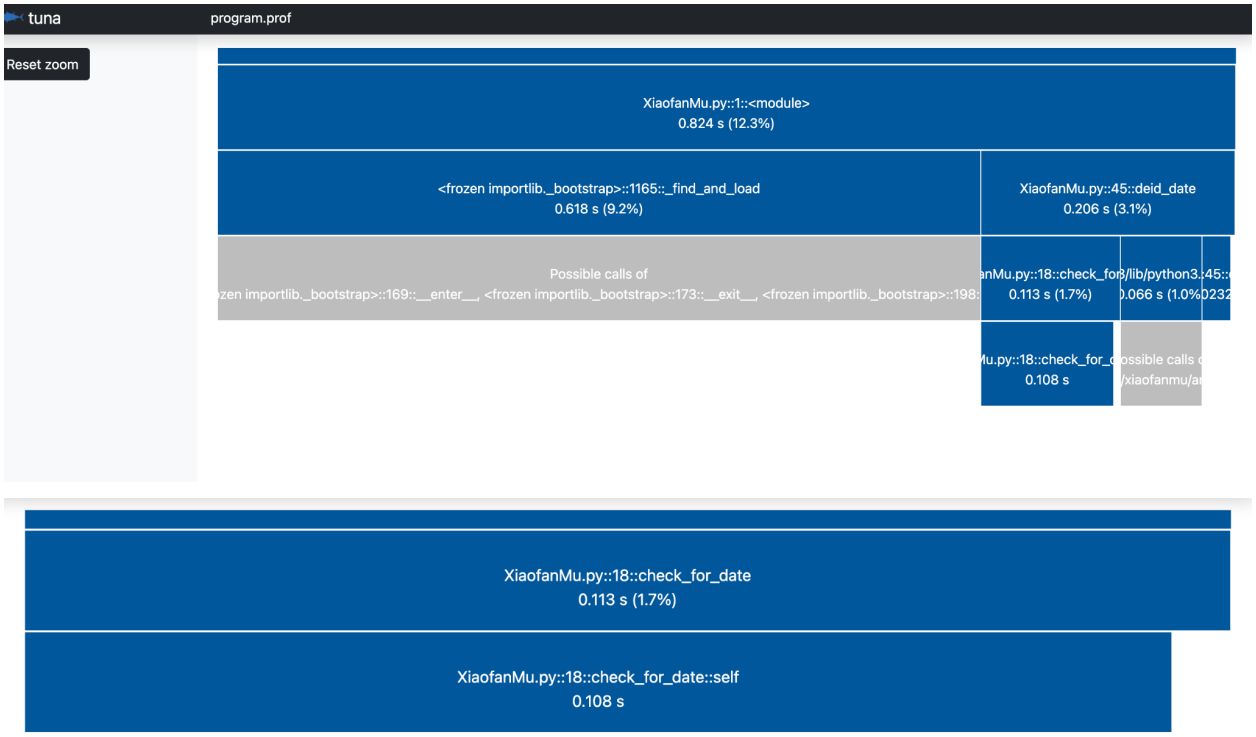
```
import re
import sys
from nltk import word_tokenize, pos_tag, ne_chunk
from nltk.tree import Tree
# we need control of how many digit each chunk should have:
date_pattern = (
    r'\b(0?[1-9]|1[0-2])/(0?[1-9]|1[0-9]|2[0-9]|3[0-1])/\d{4}\b|'  # MM/DD/YYYY
    r'\b(0?[1-9]|1[0-2])/(0?[1-9]|1[0-9]|2[0-9]|3[0-1])/\d{2}\b|'  # MM/DD/YY
    r'\b(0?[1-9]|1[0-2])/(0?[1-9]|1[0-9]|2[0-9]|3[0-1])\b|'  # MM/DD
    r'\b(0?[1-9]|1[0-2])/\d{2}\b'  # MM/YY
)



# compiling the reg_ex would save sime time!
date_reg = re.compile(date_pattern)
```

I have examined the id-phi.phrase file and found many possible patterns for the date:
MM/DD/YYYY: For example, 8/18/1989
MM/DD/YY: For example, 8/19/20.
MM/DD: For example, 8/18
MM/YY: For this one, I am not too sure, but I think it doesn't hurt the result if I put it there.
I tried to add MM/YYYY's regular expression, but the true positive rate went down, so I deleted it.
However, I found that there is a possibility that there may be something that looks like 50/50, which is not a date, but in a format of MM/DD. In order to resolve this issue, I controlled the value of the digits. For matching the day, I use (0?[1-9]|1[0-9]|2[0-9]|3[0-1]), which ensures we only match valid days (from 01 or 1 to 31). For matching the month, I use (0?[1-9]|1[0-2]), which ensures we only match valid months (from 01 or 1 to 12).
And this gave us a good result:

```
Examining "Date" category.

=========================

Num of true positives = 421

Num of false positives = 337

Num of false negatives = 61

Sensitivity/Recall = 0.873

PPV/Specificity = 0.6

=========================
```

For the profiling purpose, I used the popular Python profiling tool called “tuna”.
According to tuna, the whole program took 6.7 seconds to run.
And here are some methods that took the majority of the time when the program was running:





They are Python module, find_and_load(), died_date(), and check_for_date():
find_and_load() took 0.618 seconds, which is 9.2% of the total running time.
died_date() took 0.206 seconds, 2.1% of the total running time.
check_for_date() took 0.113 seconds(1.7%) and its self took 0.108 seconds.
From the above results, we know that our program was running efficiently, spending most of its time on expected functions and behaviors.