515 Homework 1 Question 4

Chenrui Xu

September 2021

1 Question

Algorithms are to solve problems, or more precisely, to solve problems that have a precise mathematical definition. However, in practice, figuring out what is exactly the problem is not easy at all. (You may search internet) Suppose that you would like to write an algorithm to decide the similarity between two C programs. What would you do?

2 Solution

There are two cases in this question. The first one is open sources case, and the other is black box case. Open sources case is that, we can know what the code files look like when we do the analysis. However, we cannot analyze syntactic level. On the other case, we are given two files but we cannot turn them on. The only thing we can do is getting them run.

The similarity we are talking about has a few perspectives. We can talk about the meaning level. We can talk about the functional level. We can also talk about effective level. And what is more, we can talk about structure level.

If we want to rank these levels for priority, the most important perspective should be the functions of the codes. Without any hesitation, if two code files do not have the same functions, they are not same. Of course cars and burgers will not be in the same category as they have different functions to human beings. One can drive people to somewhere else, the other keeps people away from starving.

I will talk about the black box case at first. The reason is that the methods I will use in analysing the black box case can also be used in the open sources case, but some ideas I will apply in the open sources case cannot be used in the black box case as we cannot see the original codes.

2.1 Black box case

In the black box case, the complicated situation is that we cannot know what the codes are.

Take some factory production line as an example. If an amateur people is shown around in the factory, he or she will never know how the production run if he or she has not learned the theoretical knowledge before. However, after he or she watching some raw materials, like metal, sending into the machine and some time passed, the output is a car. The man may understand that the function of the production black box is producing cars.

So the first step of the analysis is trying to figure out what are the inputs and the outputs. There are totally two sub situations here. Note that, there are actually four cases here (see Figure 1).

Input	Output
Same	Same
Same	Different
Different	Same
Different	Different

Figure 1: Four possible situations for input and output.

We cannot figure out the similarity of function if the first code requires the input of one image while the other one wants a whole sentence or we may say that the functional similarity here is zero. So, in theoretical case, we only consider controlling cases (inputs are kinds of same). We control both inputs to be same. here, when we are talking about same, it is the generalized same, which means, take computer vision as an example, if we input two cat pictures into CNN model, both outputs should be cat even though two cats are not same as inputs, but they have same category: images.

So far, I thought above ideas by myself without googling. Then, I searched online and found some knowledge about Equivalence Class Partitioning [1]. So we can use tons of data to figure out what is input and index the categories so that we can use them later.

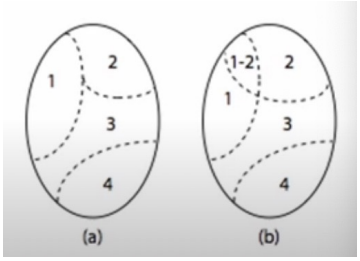

Figure 2: Categories of Inputs.

Also, we can use data for trial to get the boundary of valid data area (same mechanism with PAC learning). Once getting the data bound area, it is not hard to figure out whether two inputs are same or not. Here, we can define that same means two inputs share some joint category area.

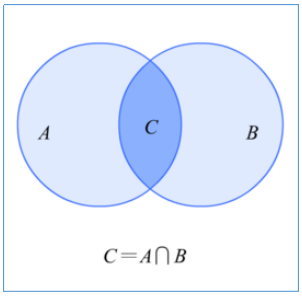

Figure 3:

Here, we use

$$S(input) = \frac{2 * C}{A + B}$$

The reason here we use 2 times C is that if not, one hundred percent overlap will only get fifty percent

value instead. And A and B can be the set of all runnable inputs. What is more, we should set a threshold (0.5 for example) here to see whether we can move on to the next step. If not, the process is over. There is no meaning of two code files if they do not have the similar inputs.

As I mentioned above, there are actually two sub situations if we do not consider inputs are different cases. We can also use the same idea to calculate the common outputs areas. If not satisfy the threshold, the process break.

Since both thresholds were passed, we will control inputs to be exactly strict same so that we can analysis the similarity of results. There are many kinds of results, like strings, numbers(int, double, etc), lists, images, etc. It is not easy for us to calculate similarity directly. So there must be some transformations. The common method is features extraction. In the area of machine learning, we also call it matrix mapping. It means that we turn all kinds of data into vectors or matrices so that we can calculate the distance between two vectors.

$$cos(\theta) = \frac{a * b}{|a| * |b|}$$

And the similarity should be

$$Similarity = cos(\theta) * S_(input)$$

Note that, a and b can be neither matrices or vectors depends on the dimension of the transformed data. The result is a number between zero (not same) and one (totally same).

Also, we can take other parameters as references. Even though we cannot see what does the code look like, when we run it via our computer, we can get many parameters from system terminal. The size of file, how much ram will it take (space complexity), the time complexity, etc. As we can only get the final parameters, we will talk about time complexity and space complexity in the open source section. However, there are some situations we cannot ignore. Given same input and same output, if the sizes of files have huge gap or the running time have significant differences, like one minute comparing one day, we will not say they have running similarity, as well as architecture.

2.2 open source case

So, in this case, we can see the source codes. Not only we can analysis the methods we mentioned above, we can also have some new methods in terms of code architecture instead of syntactic analysis. I will attribute fifty percent weight to the meaning and functional similarity and the rest fifty percent to architecture analysis.

In this step, we need to recognize some keywords. When we get one code file, we can extract some keywords like 'class','function','while','for','int', etc. There are two ways to use them. So in the following steps, we need to match some variables and modules based on the location and other information of these keywords.

On the other hand, based on these words, we will create a new structure of input data: knowledge graph. Nodes in the knowledge graph can be variables, some if/for/while judgment, some self-defined functions and some other classic functions (like 'cout', 'cin', etc).

How? When we run the code, it is not hard to extract some variables from terminal environment. Based on the time line and modules, we can know which variables belong to what function or what class.

Algorithm 1 A

SECTION 1: Run both $C1$ and $C2$;
Get $input_1$ and $input_2$ by using Equivalence Class Partitioning;
Set boundaries for $input_1$ and $input_2$;
if not same category **then**
    break
else
    $S_(input) = \frac{2*C}{A+B}$;
    if $S_(input) < 0.5$ **then**
        break
    else
        continue
    end if
end if
set $input_1 = input_2$ and get $output_1$ and $output_2$
if not vector or numbers **then**
    if image **then**
        use Convolutional kernel to extract features
    else if String **then**
        Word Embedding
    else
        other corresponding methods
    end if
else
    $cos(\theta) = \frac{a*b}{|a|*|b|}$
    $Similarity = cos(\theta) * S_(input)$
end if
SECTION 2: Check files sizes and running parameters
if significant gap **then**
    break
else
    Continue
end if
return Similarity

For example

```
def example(x):
    a=0
    for i in range(x):
        a+=i
    return a
var=10
sum=example(var)

print(sum)
```

Figure 4: Example codes

This is a sudo code, but we can view it a C-program-liked code. The function of the code is simple, just adding numbers together. Here, we got one function 'example', three variables belonging to the function. Two independent variables outside the function (or we can say, in the main body of the code), and one print function.

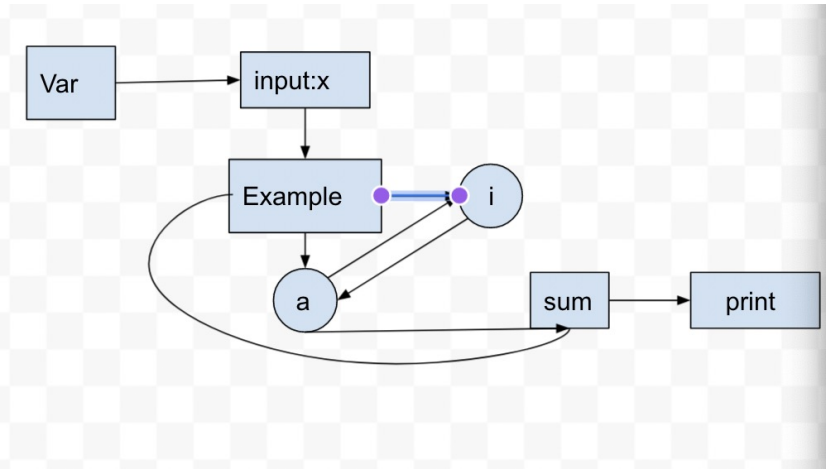

Figure 5: network

We can draw the graph like this to show the relationships between nodes. Pay attention, edges here are directed. Based on graph knowledge and network knowledge, we can turn the nodes and edges into adjacent matrix so that we can analysis numbers in the matrix then.

	var	x	example	i	a	sum	print
var	1	1	0	0	0	0	0
x	0	1	1	0	0	0	0
example	0	0	1	1	1	0	0
i	0	0	0	1	1	0	0
a	0	0	0	1	1	1	0
sum	0	0	0	0	0	1	1
print	0	0	0	0	0	0	1

Figure 6: Adjacent matrix

So, we got the matrix. Then, based on GCN knowledge, we do Fourier transform for several times to make sure both dimensions of two matrices are same.

$$H^{l+1} = \sigma(\widetilde{D}^{-\frac{1}{2}}\tilde{A}\widetilde{D}^{-\frac{1}{2}}H^l w^l)$$

And I will not mention details here and here is the reference: https://arxiv.org/pdf/1812.08434.pdf. We just need to know that we put the adjacent matrix in the formula and finally we will get a feature matrix then.

After that, we do the same operation with section one.

$$cos(\theta) = \frac{a * b}{|a| * |b|}$$

Then, we will figure out the similarity in terms of architecture. We do the algorithm one at first and section two method afterwards and give them corresponding fifty percent weight.

We can also analysis the time and space complexity here. For example, if we recognize there are nested loops in the code, we can get the running time to be $O(n^2)$.

$$S_(complexity) = \frac{\min A, B}{\max A, B}$$

This may take ten percent of the architecture part.

Algorithm 2 B

SECTION 1:Run Alg A at first
give similarity part one 50%

SECTION 2: Recognize key words
Construct network graph based on key words
Turn network graph into adjecent matrix
Fourier transformation and require the same dimensions

$$cos(\theta) = \frac{a * b}{|a| * |b|}$$

OR other distance methods
give this part 45% weight

SECTION 3: calculate time/space complexity and other parameters based on

$$S_(complexity) = \frac{\min A, B}{\max A, B}$$

give them 5% weight

add all kinds of similarities together
return Similarity

3 References

1.https://asset-pdf.scinapse.io/prod/2334860424/2334860424.pdf

2.https://arxiv.org/pdf/1812.08434.pdf