**Institution : Kisii University (main campus)**

**Unit : Special Topics (soen 320)**

**Name : Bakari Mtua Kilu**

**Reg No : IN16/00001/18**

**Session : Y3 S1**

**Assignment 7**

**0797544060**

**10/12/2021**

Review the IEEE Computing edge magazine (Sept 2020) publication and attempt to the following question

1. Humans are avid consumers of visual content. Every day, people watch videos, play games, and share photos on social media. However, there is an asymmetry - while everybody is able to consume visual data, only a chosen few are talented enough to express themselves visually. For the rest of us, most attempts at creating realistic visual content end up quickly “falling off” what we could consider to be natural images. Review and investigate available machine learning approaches for preserving visual realism while creating and manipulating photographs.

-The following are the approaches for preserving visual realism while creating and manipulating photographs:-

* Casting doubt approach

-Consider how we may try to determine how often different types of pointer arithmetic are used in a large code base. If you’re brave enough to try to express this in one of the customization languages of the leading static analysis tools, you are a better person than I. Here’s how I would do it with a Cobra query that can both be written and executed in a few seconds:

$ cobra −pat “\\* ) ( .\* + .\* ) ” \*.c

The pattern I used attempts to match type casts, enclosed in round braces, that end with a star operator, indicating that whatever follows is cast to a pointer. Next, the pattern checks that the expression in round braces that follows contains a plus operator, indicating that an addition is being performed. The syntax for the pattern expression we used here is a simplified version of a regular expression. The initial star is escaped so that it cannot be misinterpreted as a metasymbol. The dot matches any token text, and the combination is used to indicate an arbitrary sequence of tokens. Cobra makes sure that the round braces in the pattern match, which forces the text in between the braces to refer to a single expression. In an interactive session with the Linux source code, matches to this query pattern are found in roughly 1 s on a multicore system. If you’re curious, there are approximately 6,000 such matches in the 14.9 million lines of Linux code. Here’sthe first one, found in file mm/slub.c, online 247:

return \*(void \*\*)(object + s−>offset);

* Fast approximation approach

-We can get the tool to work fast because the matching algorithm is simple and, therefore, trivial to parallelize. It can, in principle, be executed independently on each separate file, so given enough CPUs, the 21,987.c files in the Linux distribution we looked at could be scanned within a fraction of a second. On my own 32-core system, the processing takes roughly 20s, which includes approximately 9 s for preparing the lexical token stream itself. I should add that this last code-mining method I used is not precise. For example, I did not bother to distinguish among the three different parts of the for-loop. The final set of matches was small enough, even for a code base of this size, that I could afford to overshoot the final set by a small amount and then peruse the matches for the little gems I was really seeking.

* Code as text

-It would be a little more useful if our algorithm could deduce that a name, when it is followed by an open round brace and preceded by a word other than “void” (in “good” programs, e.g., in the definition of a function), will almost never be preceded by either a colon or a semicolon when it appears elsewhere in the code (again, followed by an open round brace). That is, the value returned by the function is effectively used by the caller in an expression or an assignment.

As you can see, trying to describe these code patterns as just textual coincidences leads to some tortured language, but the statistics should definitely bear out these types of correlations. Whether they are also discoverable by a machine-learning algorithm without further guidance is another matter. I have my doubts. Unfortunately, it is generally easy to record patterns that are present but much harder to record patterns that are rare or absent. An example of a well-known pattern that is absent from well-written English prose, for instance, is that a preposition is rarely followed by a period. Our teachers made sure that we all follow that rule, at least most of the time.