Towards Open Natural Language Feedback Generation for Novice Programmers using Large Language Models

Charles Koutcheme, charles.koutcheme@aalto.fi
Aalto University

Context

This poster proposes an approach for automatically generating programming feedback. The project idea builds on the intuition of a teaching assistant (TA) helping a student who is stuck.

Project Idea

When shown a student's program, the TA:

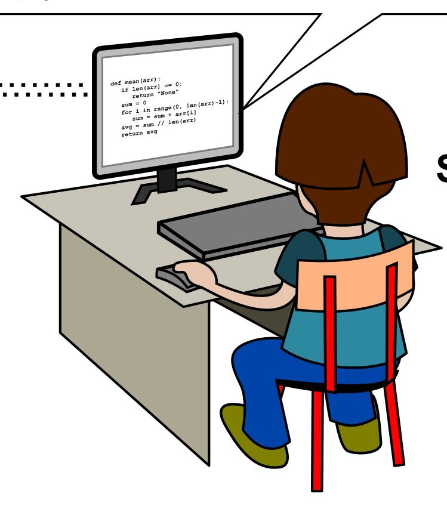
1) Envisions a working solution to the given problem that is close to the student's code and the changes that need to happen to transform the student's code into the pictured working solution.

2) Then, the TA comes up with a description of what is happening in the current incorrect program and highlights what makes it incorrect.

The natural language description of what makes the code incorrect and the required changes to arrive at the correct solution are the TA's feedback to the student.

Our idea builds from the intuition that we can map this human thought process to existing machine-learning approaches leveraging open large language models (LLMs).

Write a function "mean", that takes as an argument a list of numbers and computes the average of its elements.
If the list is empty, the function returns the null value.

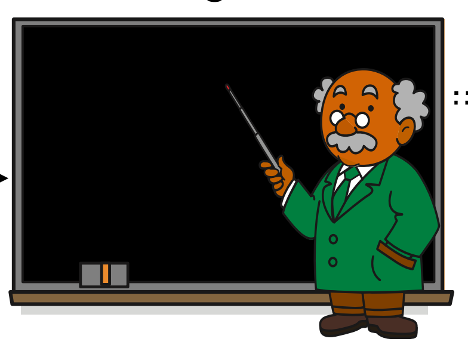


Envisions a working solution
Describes the incorrect parts

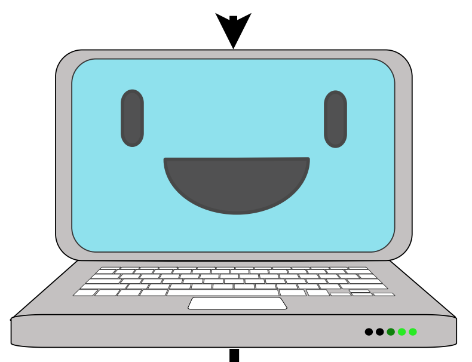
student's incorrect solution

```
1. def mean(arr):
2.    if len(arr) == 0:
3.       return "None"
4.    sum = 0
5.    for i in range(0, len(arr)-1):
6.       sum = sum + arr[i]
7.    avg = sum // len(arr)
8.    return avg
```

Path A.
Human feedback



Path B.
The proposed approach



Program repair

Recent generative models are capable of repairing code.

```
1. def mean(arr):
2.    if len(arr) == 0:
3.       return "None"
4.    sum = 0
5.    for i in range(0, len(arr)-1):
6.       sum = sum + arr[i]
7.    avg = sum // len(arr)
8.    return avg
```

Program repair

```
1. def mean(arr):
2.    if len(arr) == 0:
3.       return None
4.    sum = 0
5.    for i in range(0, len(arr)):
6.       sum = sum + arr[i]
7.    avg = sum / len(arr)
8.    return avg
```

Line by line code explanation

Large Language Models can also write line-by-line descriptions of what is happening in students' code.

Approach understanding

By modelling code and natural language jointly, we could come back with the approach taken by the student to solve the problem.

Program understanding

Line by line code explanation

```
(2) If the length of the list is 0,
(3) the function returns the string "None".
(4) Initialises a summation variable.
(5) Loops through the array from the
    first element up until before the last one...
(6) ...and computes the sum of these elements.
(7) Divides the sum by the length of the array,
    and keeps the integer part.
(8) Returns the computed value.
```

```
1. def mean(arr):
2.    if len(arr) == 0:
3.       return "None"
4.    sum = 0
5.    for i in range(0, len(arr)-1):
6.       sum = sum + arr[i]
7.    avg = sum // len(arr)
8.    return avg
```

Approach understanding

If the list is empty, return "None".	EMPTY LIST (lines 2–3)
Sum all elements of the array, except the last one.	SUMMATION (lines 4–6)
Computes the integer division of the sum by the length of the array.	AVERAGING (line 7)

```
1. def mean(arr):
2.    if len(arr) == 0:
3.       return None
4.    sum = 0
5.    for i in range(0, len(arr)):
6.       sum = sum + arr[i]
7.    avg = sum / len(arr)
8.    return avg
```

Line by line code explanation

```
(2) If the length of the list is 0,
(3) the function returns None.
(4) Initialises a summation variable.
(5) Loops through all elements of the array...
(6) ...and computes the sum of these elements.
(7) Divides the sum by the length of the array.
(8) Returns the computed value.
```

Approach understanding

EMPTY LIST (lines 2–3)	If the list is empty, returns the null value.
SUMMATION (lines 4–6)	Sums all elements of the array.
AVERAGING (line 7)	Divides the sum by the length of the array.

Generating feedback

From **Program Understanding**, we get the root of the problems

From **Program Repair**, we get the action to perform to correct the code

Message

```
In case of an empty list,
the program does not return None.

When doing the summation, the program
does not loop up until the end of the array.

When computing the mean,
the program does not compute the full average.
```

Hint

```
On line 2, replace "None" by None
On line 5, remove the -1
On line 7, replace the integer division ("//")
operator with the division operator ("/")
```

The message

The message part of the feedback will be formatted by an analysis of the differences between the code descriptions of the correct and incorrect solutions.

The hint

We obtain the action to perform using bug localisation (during programer repair).

Aalto University
School of Science

Scan me