1 Correctly Predicted Examples

input	call the method CLASS_0 . FUNC_0 with an argument boolean True .
pred.	CLASS_0.FUNC_0(True)
input	ANY_1 is an string STR_0 formatted with CLASS_0 . ANY_0 and result of the method CLASS_0 . FUNC_0 , respectively .
pref.	ANY_1 = 'STR_0' % (CLASS_0.ANY_0, CLASS_0.FUNC_0())
input	remove last element for ANY_0 .
pred.	ANY_0 = ANY_0[:-1]
remark	The fact that the model learned to map the token ``last'' in the query to ``-1'' in code, tells us that the model has learnt to translate some semantics in the input query to code.

2 Incorrectly Predicted Examples

input	ANY_0 is an tuple with 3 elements : None , result of method CLASS_0 . FUNC_0 and None .
ref.	ANY_0 = None, CLASS_0.FUNC_0(), None
pred.	ANY_0 = None, (None,), CLASS_0.None
remark	A typical form, XXX.YYY appears in code, whenever the user wants to express that YYY is an attribute of entity. Here, we hypothesize that the model assigned ``None'' as CLASS_0's attribute due to the period at the end of the sentence.
input	call the function CLASS_0 . FUNC_0 -LSB- CLASS_0 . FUNC_0 -RSB- with 3 arguments : raw string STR_0 , ANY_2 and ANY_0 without the first and last element
ref.	ANY_1 = CLASS_0.FUNC_0('STR_0', ANY_2, ANY_0[1:-1])
pred.	ANY_0 = CLASS_0.FUNC_0('STR_0', ANY_0 + 1)
remark	We observe that the model typically performs well on short phrases, but on long phrases it fails to capture all the nuances expressed in different parts of the sentences. In the following example, the model completely fails to realise ANY_0 as a list

3 Interseting finds

input	ANY_0 is an empty dictionary .
ref.	ANY_0 = dict()
pred.	ANY_0 = {}
remark	Here, the model has correctly predicted an empty dictionary, but the evaluation has missed this.
input	ANY_0 is an list with an element ANY_1 .
ref.	ANY_0 = [ANY_1]
pred.	ANY_0 = [ANY_1]
remark	This is the only example where the model has correctly predicted on a list based query. We hypothesize that this might be because the list based queries are generally very long causing the model to forget long term dependencies in the code.
input	substitute ANY_1 for ANY_0 , ANY_4 , ANY_3 and ANY_2 , respectively .
ref.	ANY_0, ANY_4, ANY_3, ANY_2 = ANY_1
pred.	ANY_0, ANY_4, ANY_3, ANY_2 = ANY_1
remark	It is interesting to notice that the model is able to understand which part of the code fall on the left and right side of the assignment operator. Also, it correctly places all the variables in their proper order.