EMNLP 2022

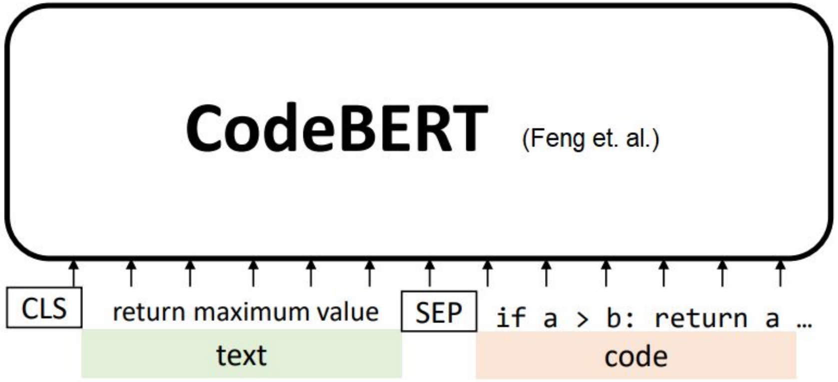
Benchmarking Language Models for Code Syntax Understanding

Da Shen[1], Xinyun Chen[2†], Chenguang Wang[3†], Koushik Sen[4], Dawn Song[4]

[†]Corresponding authors
[1]University of Maryland, College Park, [2]Google Research, Brain Team
[3]Washington University in St. Louis, [4]University of California, Berkeley

Pre-trained language models can understand code

- Represent the input as a token sequence without explicitly modeling its structure.
- Impressive performance in both NLP and program understanding.

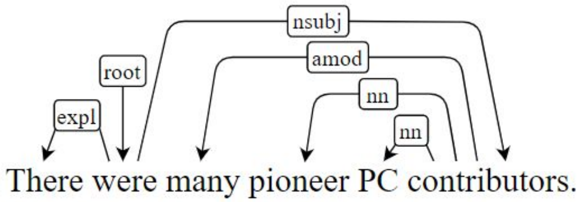


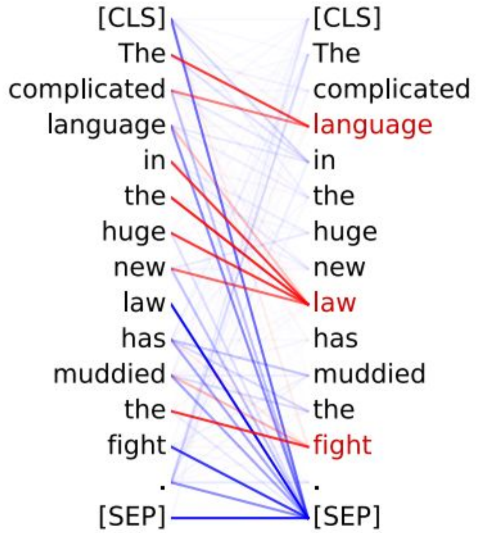
Results on code search		
Model	PYTHON	JAVA
CNN	57.1	52.7
BIRNN	32.1	28.7
ROBERTA	80.1	66.6
CODEBERT	**86.9**	**74.8**

What is the reason behind strong understanding?

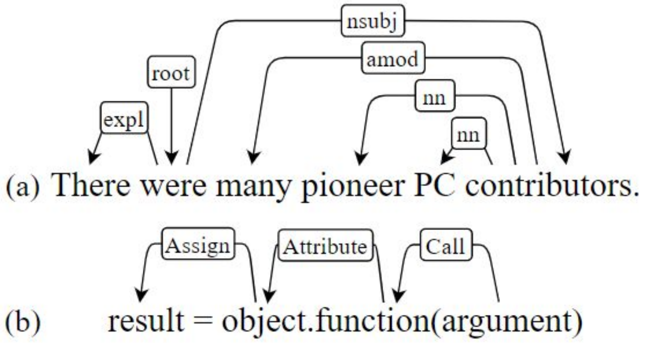
- Language models also achieve good results on natural language understanding tasks.

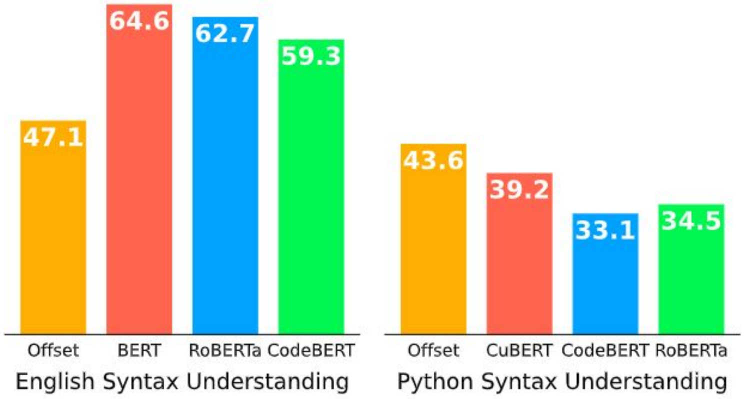
- One of the reasons: attention heads learn to capture natural language syntax during pre-training.





Does pre-training capture programming language syntax?





- We create the CodeSyntax dataset to benchmark pre-trained models for identifying the syntactic structures of programs.
- Key findings: pre-trained code language models even perform worse than simple offset baselines on code syntax understanding tasks.

Our CodeSyntax dataset

- For code syntax understanding.
- Python and Java source code from CodeSearchNet.
- Each code sample is an entire function.
- Annotated with syntactic relationships between tokens.

Head node

Dependent node

Relation	Code Example	
head→dependent	Python	Java
Assign: target→value	**target** = **10**	int **target** = **10**;
Call: func→args	**function**(**arg**)	**function**(**arg**);

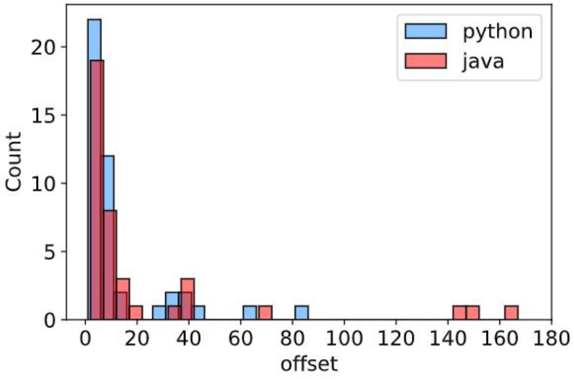
For:
for→body
	for target in iter:
	    body
		for (initializers;
		    test; updaters) {
		    body;
		}

If:
if→else
	if condition:
	    body1
	else:
	    body2
		if (condition) {
		    body1;
		} **else** {
		    body2;
		}

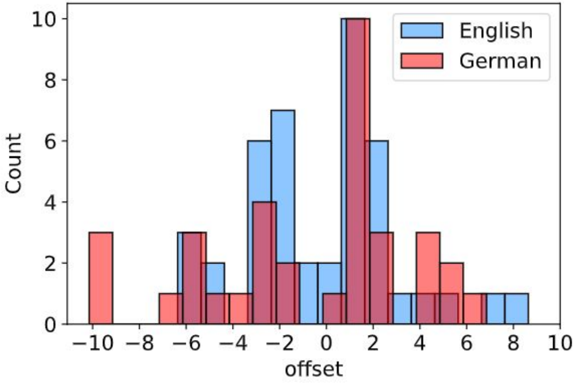
If:
if→body
	if condition:
	    body1
	else:
	    body2
		if (condition) {
		    body1;
		} else {
		    body2;
		}

Our CodeSyntax dataset

- Compared to natural language dependencies, the relation edges in the programs tend to connect tokens much farther away from each other.



(a) CodeSyntax.



(b) Natural language corpus.

Results

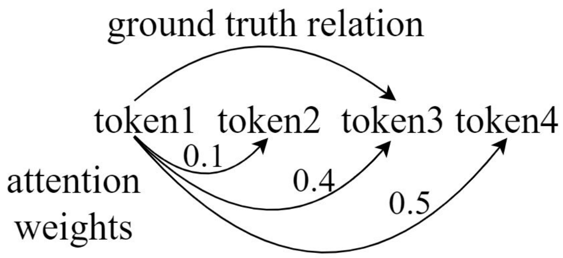
- Surprisingly, attention heads do not effectively capture code syntax.

- RoBERTa vs. CodeBERT:
  Pre-training on a large-scale code corpus, in addition to natural language corpus, does not yield a notably better understanding of code syntax.

Language	Model	Top-k Score k=1	k=3	k=10	k=20
Python	Offset	43.6	63.7	87.3	94.9
	Keyword	15.7	21.9	23.6	23.8
	Combined	**49.4**	**69.7**	**90.1**	**96.3**
	CuBERT	39.2	58.4	81.3	91.4
	CodeBERT	33.1	51.8	78.6	89.2
	RoBERTa	34.5	56.9	82.5	91.3
Diff (Model - Baseline)		-10.2	-11.3	-8.8	-4.9
Java	Offset	52.7	71.5	87.1	94.3
	Keyword	22.4	27.3	30.2	30.6
	Combined	**60.4**	**77.2**	**90.0**	**96.1**
	CuBERT	39.7	59.8	80.0	90.2
	CodeBERT	36.3	57.1	78.3	88.8
	RoBERTa	34.7	57.8	80.3	90.5
Diff (Model - Baseline)		-20.7	-17.4	-10.0	-5.9

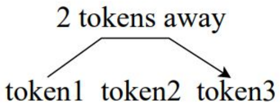
Evaluation

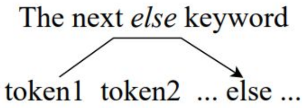
- **Code Models:** CuBERT and CodeBERT



- **Metric:** Top-k scores. Given a head token, the prediction is correct if the attention weight over the dependent token is among the top-k highest.

- **Baselines**:
  - Offset Baseline with fixed offset *i*.
  - Keyword baseline with fixed keyword *key*.





Case studies

- Attention is highly capable of performing keyword matching.
- When the head and dependent tokens are diverse, it is challenging for attention.
- Attention can not effectively utilize the relative positions of tokens to learn the relations, even if the tokens are nearby.

Relation	Score		Offset	Diff
	CuBERT	Offset		
If:if→else	92.7	5.7	17	87.1
If:body→orelse	29.2	7.1	12	22.0
If:if→body	31.5	23.1	7	8.4
For:for→body	30.4	32.7	7	-2.3
Assign:target→value	39.8	71.2	2	-31.4
While:test→body	16.2	48.5	4	-32.4
Call:func→args	59.3	93.2	2	-33.9

Conclusion

- Programming languages have hierarchical structures, long-term dependencies, and frequent keywords.
- For code syntax understanding, the pre-trained models even perform worse than simple baselines, and often attend to frequent nearby tokens regardless of hierarchical code structure.
- Designing new architectures and pre-training algorithms to leverage code structures are important future work for code learning.

Thank you!

Code: https://github.com/dashends/CodeSyntax