

# **Pre-trained Models and Benchmark for Code Intelligence**

(from the perspective of an NLP researcher)

Nan Duan (段楠)

Principal Researcher & Research Manager

Natural Language Computing Group

Microsoft Research Asia

Keynote at KDD workshop on Programming Language Processing (PLP), 2021-08-15

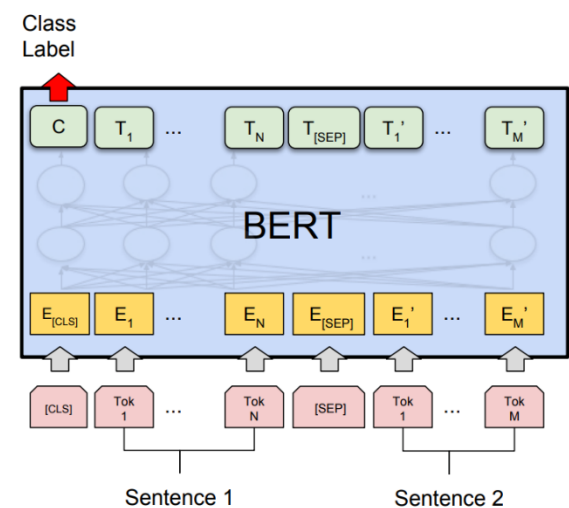
# Acknowledge

- I'd like to thank my colleagues/students that contributed to this code intelligence project, including Duyu Tang (Tencent), Shuai Lu (Microsoft), Daya Guo (Sun Yat-sen University), Junjie Huang (Beihang University), Shuo Ren (MSRA), Zhangyin Feng (Harbin Institute of Technology), Long Zhou (Microsoft), Shujie Liu (Microsoft), Ambrosio Blanco (Microsoft), Ming Zhou (Sinovation), Lidong Zhou (Microsoft), Ming Gong (Microsoft), Linjun Shou (Microsoft), Daxin Jiang (Microsoft), Alexey Svyatkovskiy (Microsoft), Shengyu Fu (Microsoft), Colin Clement (Microsoft), Shao Kun Deng (Microsoft), Dawn Drain (Microsoft), Michele Tufano (Microsoft), Neel Sundaresan (Microsoft), Marc Brockschmidt (Microsoft), Miltiadis Allamanis (Microsoft).

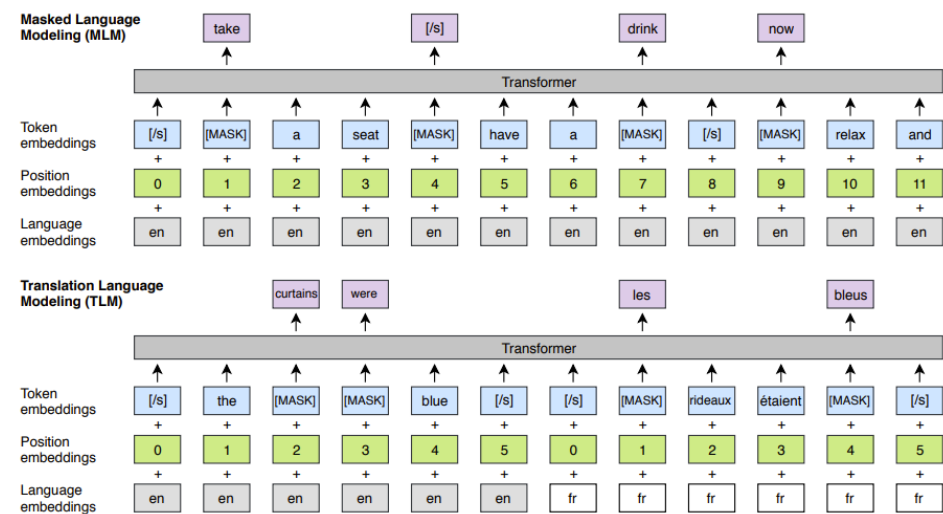
# Today's Agenda

- **Background**
- Pre-trained Models for Code Intelligence
- Benchmark for Code Intelligence
- Conclusion & Future Work

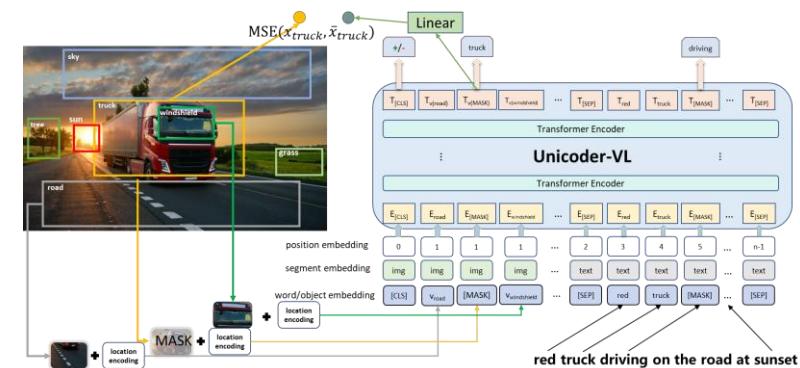
# Current NLP Paradigm: Large-scale Pre-trained Models



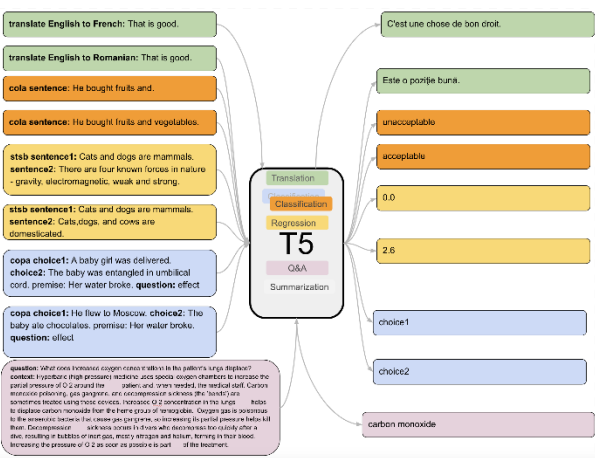
**BERT** (Devlin et al., 2018)



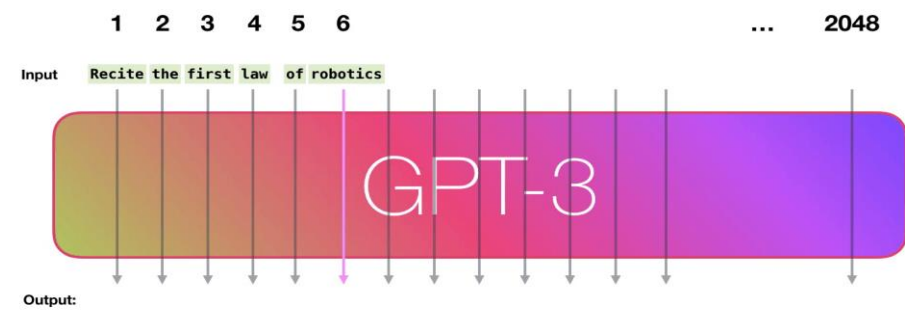
**XLM** (Lample and Conneau, 2019)



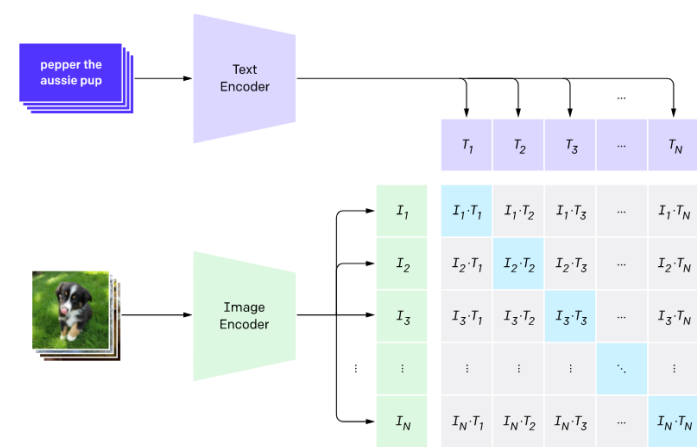
**Uniconer-VL** (Li et al., 2020)



**T5** (Raffel et al., 2020)



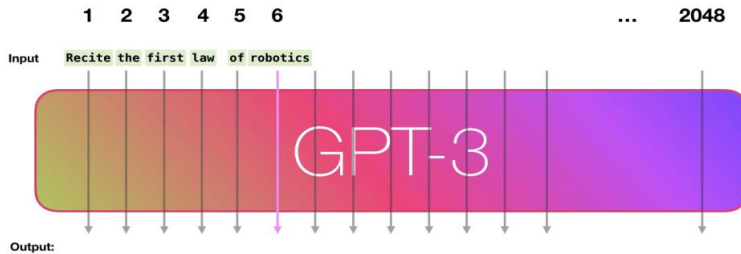
**GPT-3** (Brown et al., 2020)



**CLIP** (Radford et al., 2021)

# Self-supervised Learning in Language Pre-training

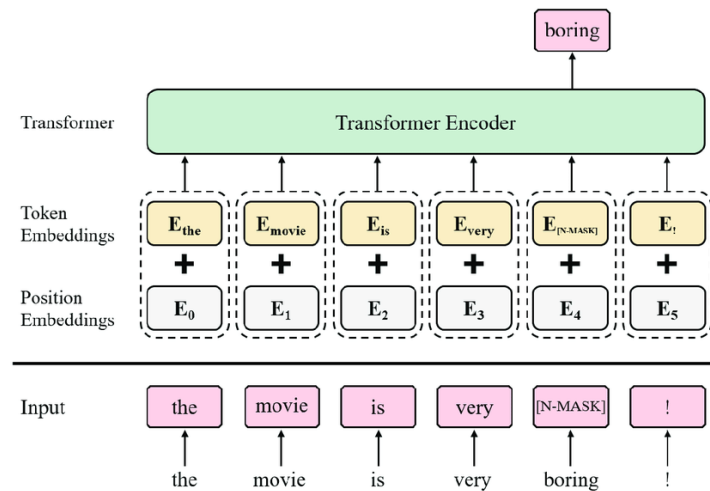
## Auto-regressive Decoding



GPT-3 (Brown et al., 2020)

maximize the likelihood under the forward auto-regressive factorization

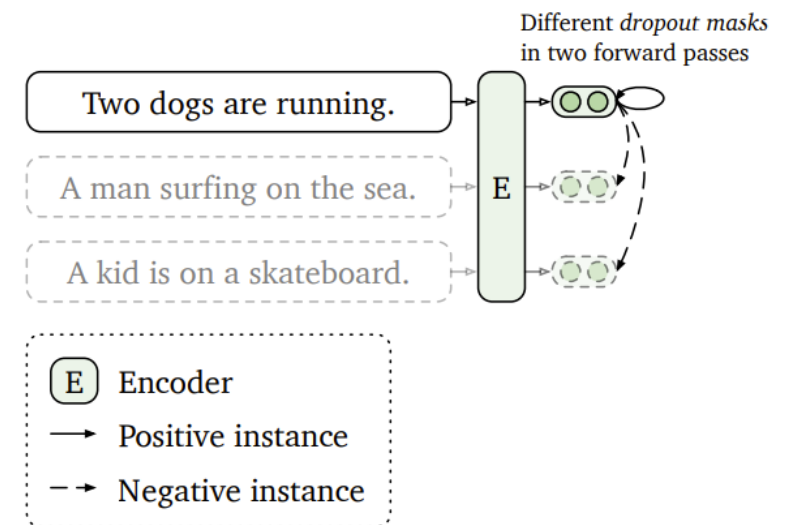
## Denoising Auto-encoding



BERT (Devlin et al., 2018)

reconstruct masked words/spans/sentences from corrupted inputs

## Contrastive Learning

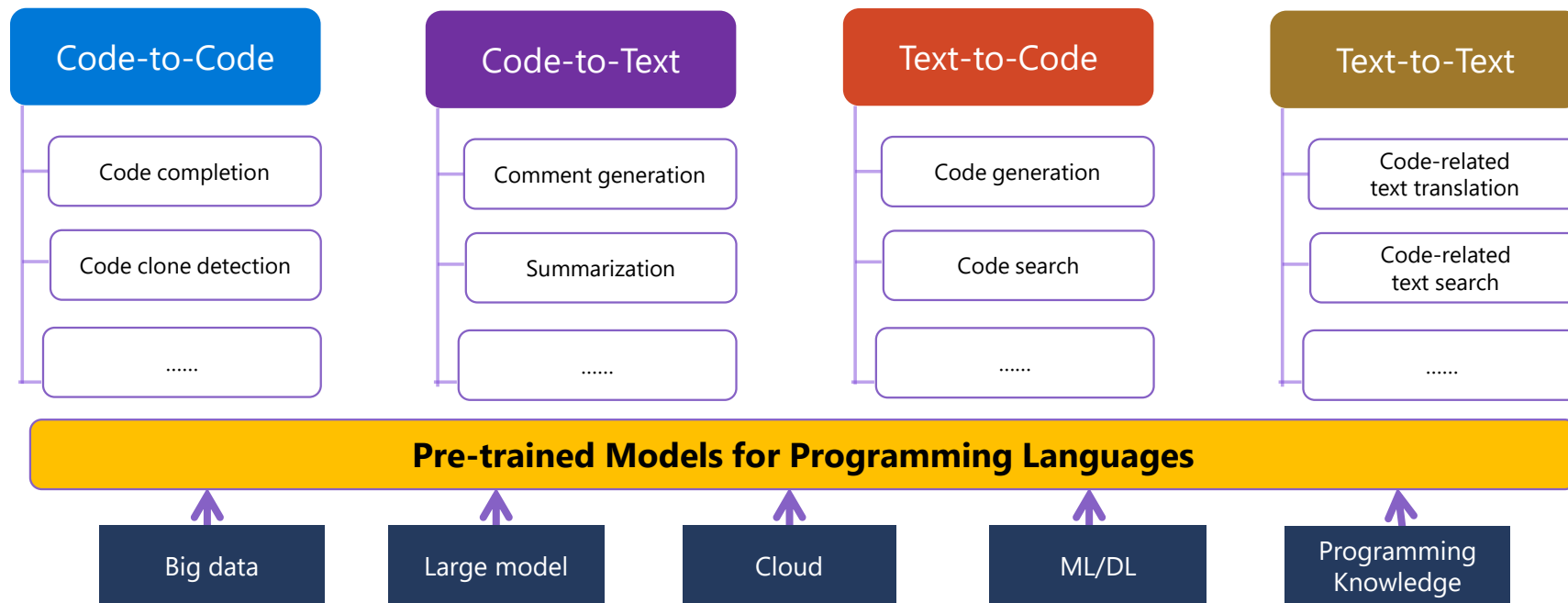


SimCSE (Gao et al., 2021)

learn representations such that similar samples stay close to each other

# Large-scale Pre-training for Code Intelligence

To build large-scale pre-trained models for code to help developers to improve their programming productivity.

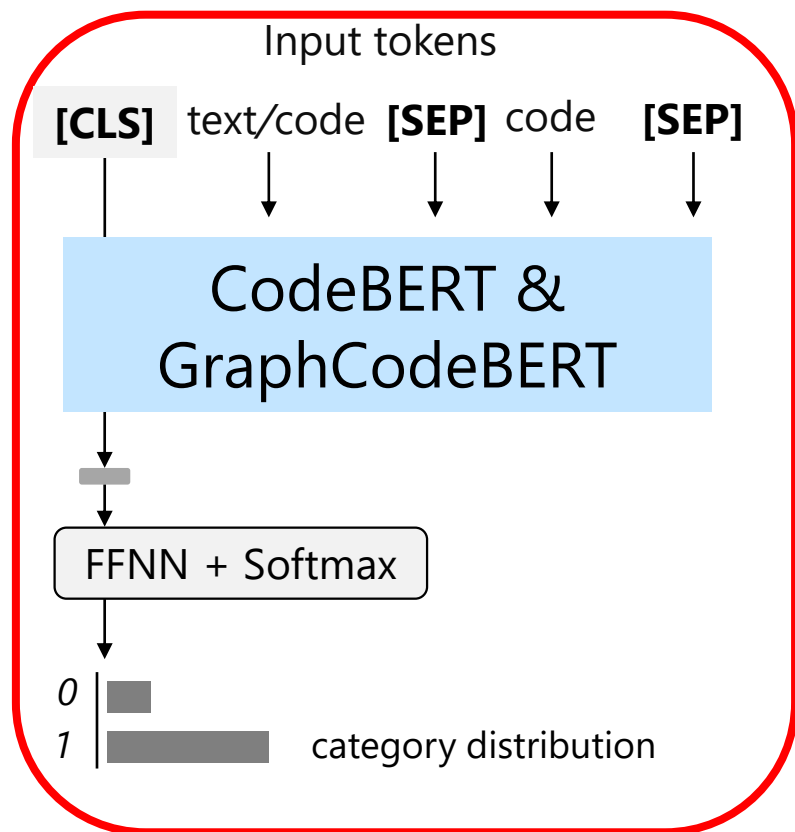


# Today's Agenda

- Background
- **Pre-trained Models for Code Intelligence**
- Benchmark for Code Intelligence
- Conclusion & Future Work

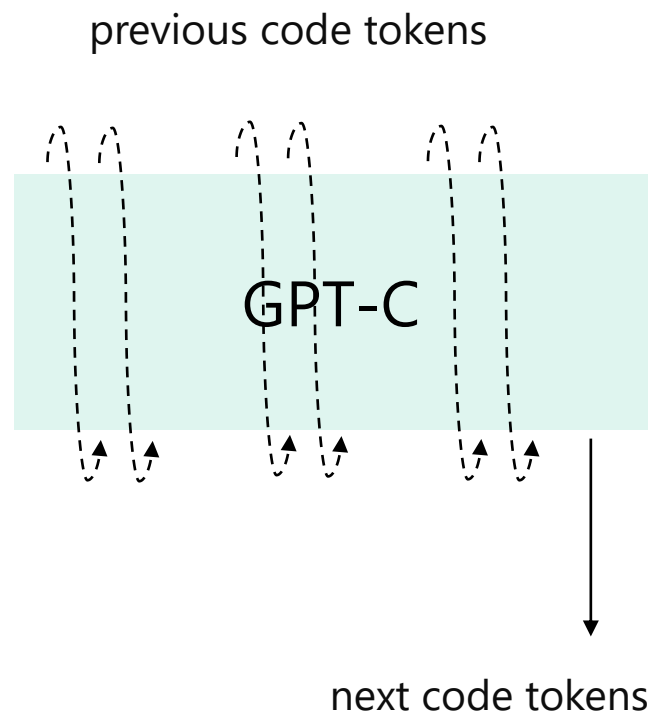
# Three Pre-trained Models for Code

## Understanding

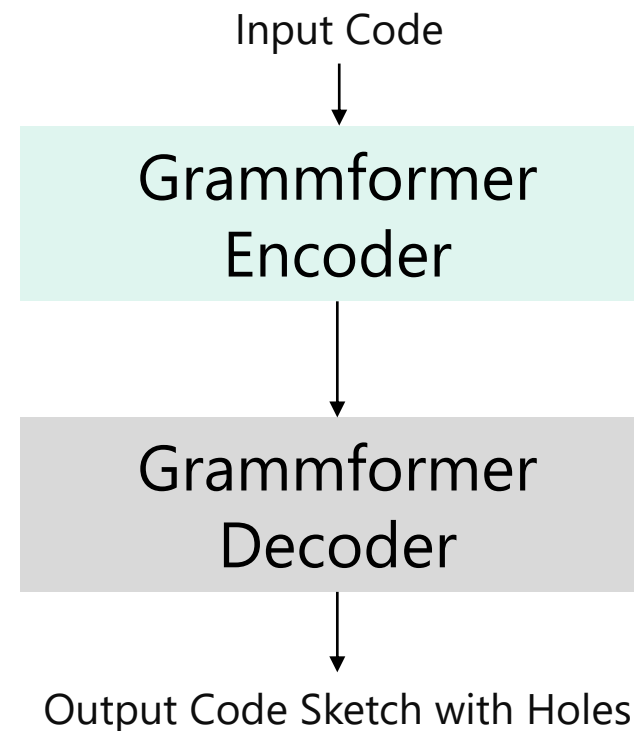


Accepted by EMNLP 2020 and ICLR 2021.

## Generation



Submitted to EMNLP 2021.



Submitted to NeurIPS 2021.

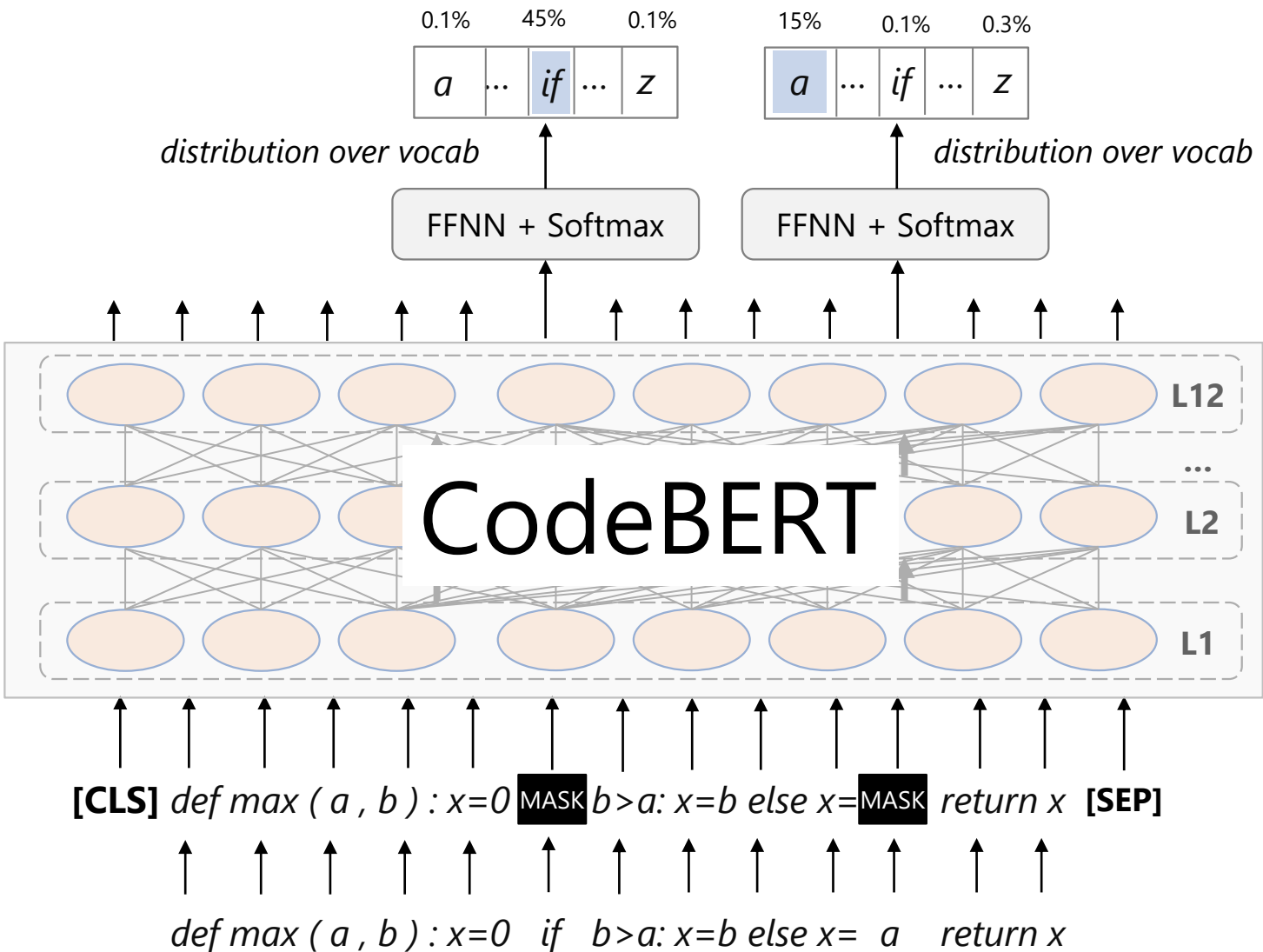


# CodeBERT: Pre-Train with Code

Predict the masked code token with the output of CodeBERT

## Source code

```
def max(a, b):
    x=0
    if b>a:
        x=b
    else:
        x=a
    return x
```



Output

Transformer

Randomly mask  
15% of tokens

Input

# CodeBERT: Pre-Train with Code+Text

Predict the masked  
code/text tokens with the  
output of CodeBERT

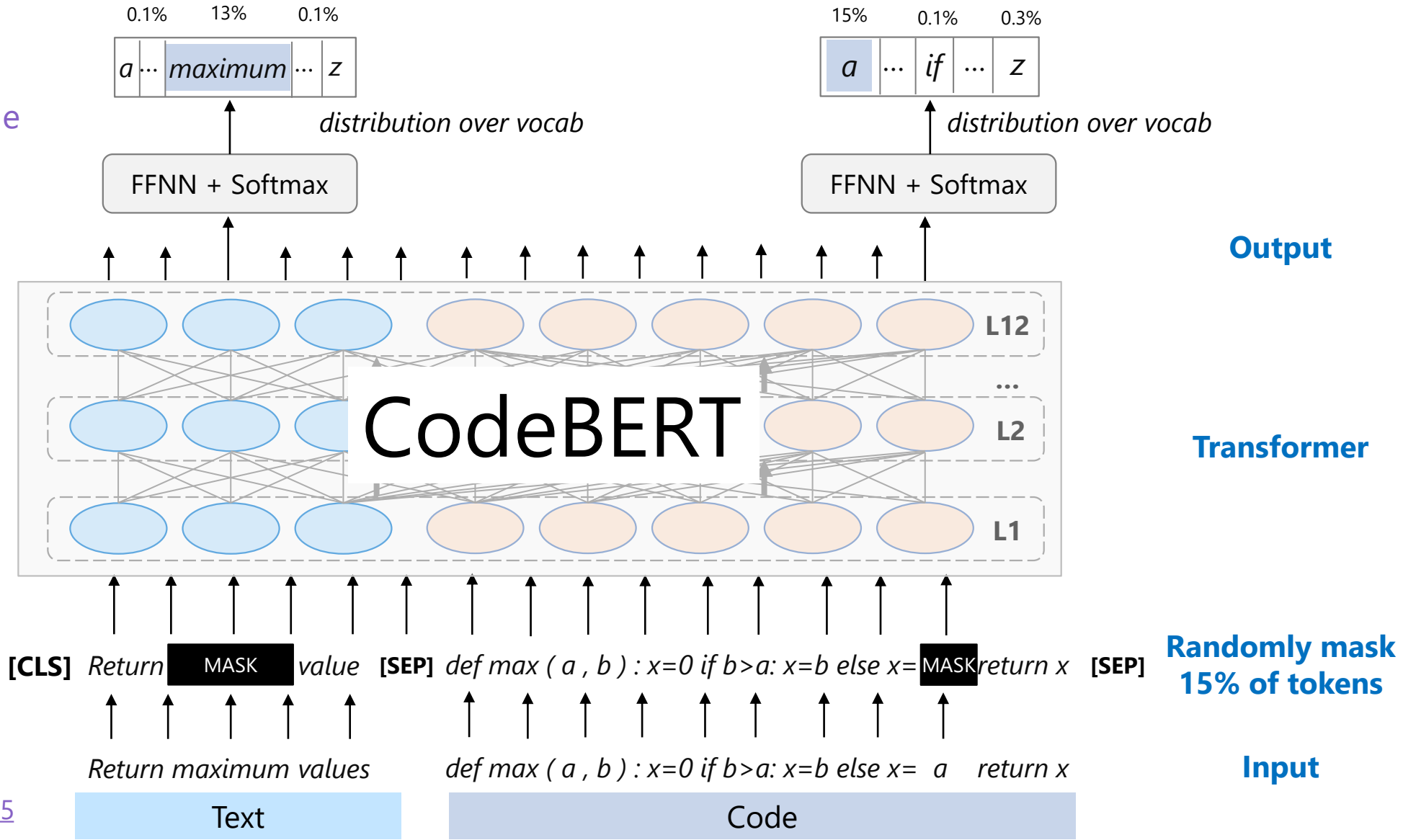
Source code

```
def max(a, b):  
    x=0  
    if b>a:  
        x=b  
    else:  
        x=a  
    return x
```

Comment

Return maximum value

<https://arxiv.org/abs/2002.08155>



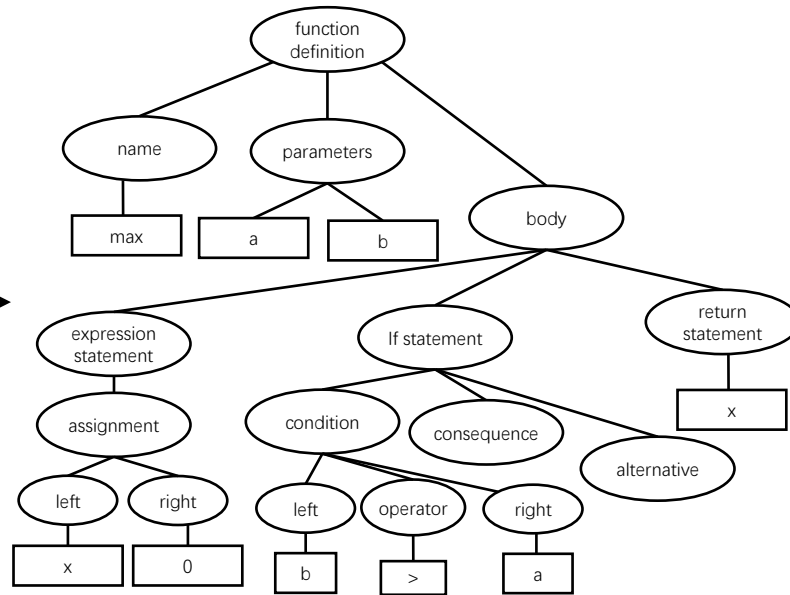
# Code Structure

## Source code

```
def max(a, b):  
    x=0  
    if b>a:  
        x=b  
    else:  
        x=a  
    return x
```

TreeSitter  
(public tool)

## Parse into AST



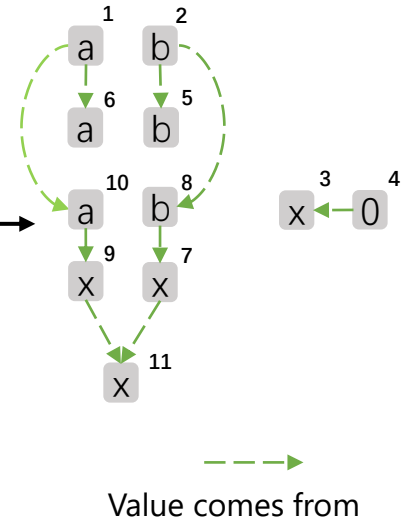
Identify variable  
sequence in AST

## Identify variable sequence

```
def max(a1, b2):  
    x3=04  
    if b5>a6:  
        x7=b8  
    else:  
        x9=a10  
    return x11
```

Extract variable relationship  
from AST according to  
paths between variables

## Variable relationship



# GraphCodeBERT: Pre-Train with Code+Text+Structure

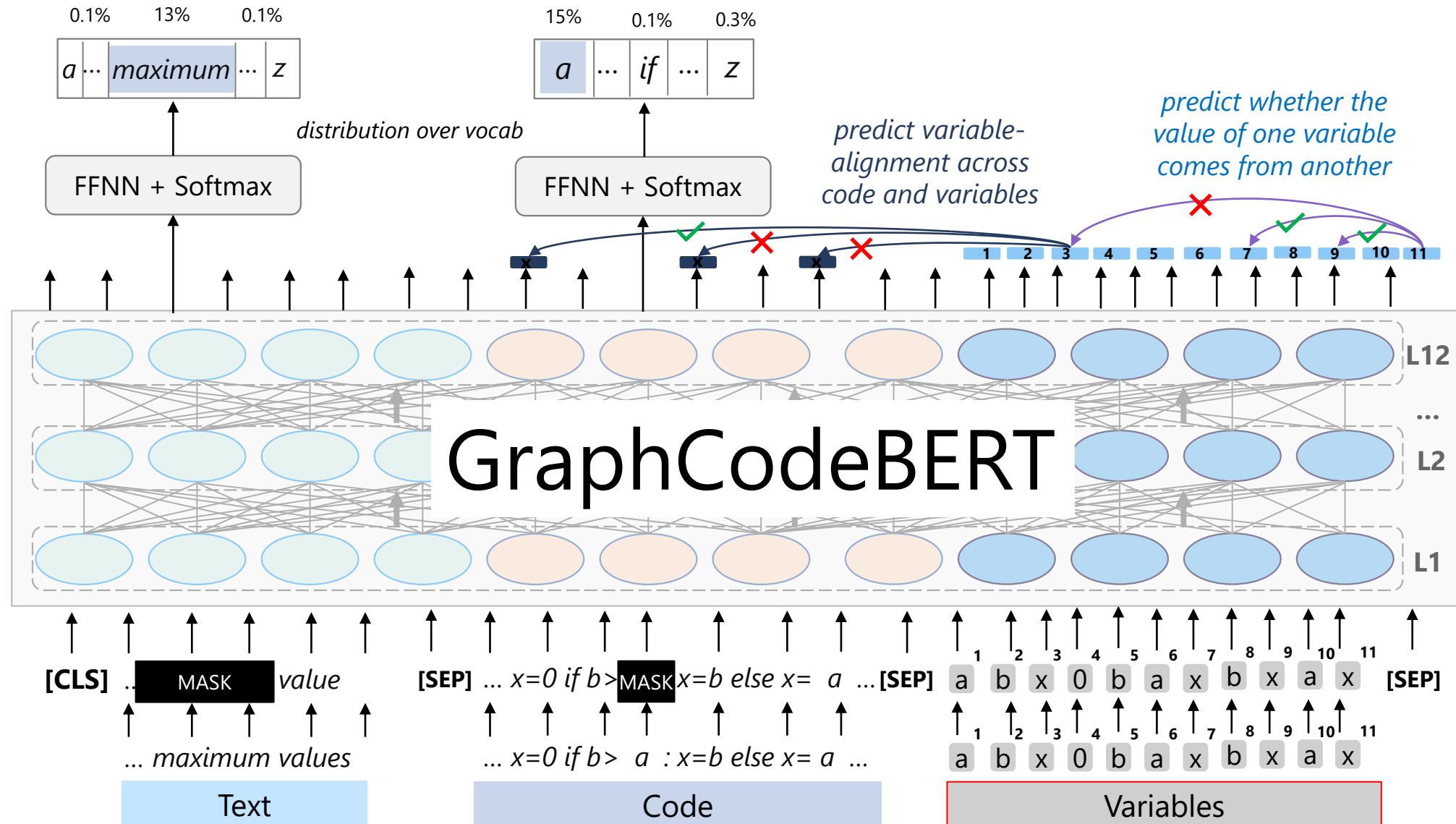
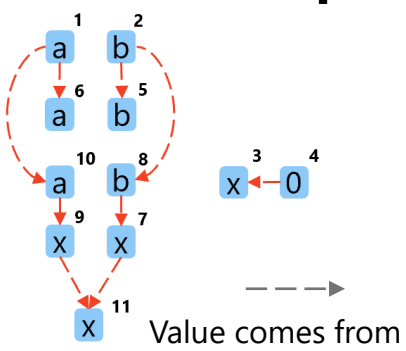
## Source code

```
def max(a, b):
    x=0
    if b>a:
        x=b
    else:
        x=a
    return x
```

## Comment

Return maximum value

## Variable relationship



# Understanding Results

Results on code search.

Model	Ruby	JavaScript	Go	Python	Java	PHP	Overall
BiRNN	0.213	0.193	0.688	0.290	0.304	0.338	0.338
RoBERTa	0.587	0.517	0.850	0.587	0.599	0.560	0.617
RoBERTa (code)	0.628	0.562	0.859	0.610	0.620	0.579	0.643
CodeBERT	0.679	0.620	0.882	0.672	0.676	0.628	0.693
GraphCodeBERT	<b>0.703</b>	<b>0.644</b>	<b>0.897</b>	<b>0.692</b>	<b>0.691</b>	<b>0.649</b>	<b>0.713</b>

**Input:**

How to read text file in Python?

**Output:**

```
def read_text_file(filename, encoding="utf-8"):
    """
    Reads a file under python3 with encoding (default UTF-8).
    Also works under python2, without encoding.
    Uses the EAFP (https://docs.python.org/2/glossary.html#term-eafp)
    principle.
    """
    try:
        with open(filename, 'r', encoding) as f:
            r = f.read()
    except TypeError:
        with open(filename, 'r') as f:
            r = f.read()
    return r
```

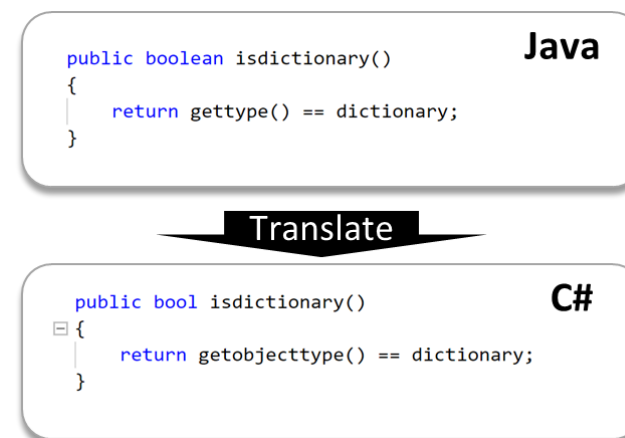
code source from [GitHub](#)

<https://github.com/microsoft/CodeBERT>

# Generation Results

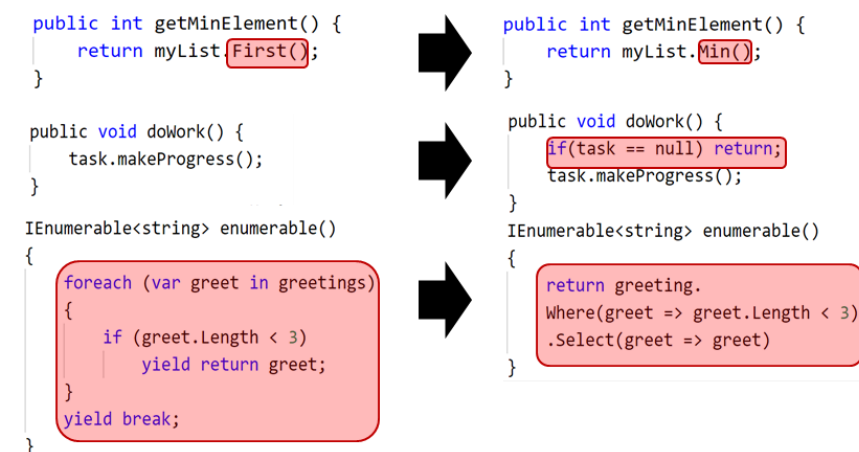
Results on code translation.

Model	Java->C#		C#->Java	
	BLEU	Accuracy	BLEU	Accuracy
Naïve	18.54	0.0	18.69	0.0
PBSMT	43.53	12.5	40.06	16.1
RoBERTa (code)	77.46	56.1	71.99	57.9
CodeBERT	79.92	59.0	72.14	58.0
GraphCodeBERT	<b>80.58</b>	<b>59.4</b>	<b>72.64</b>	<b>58.8</b>



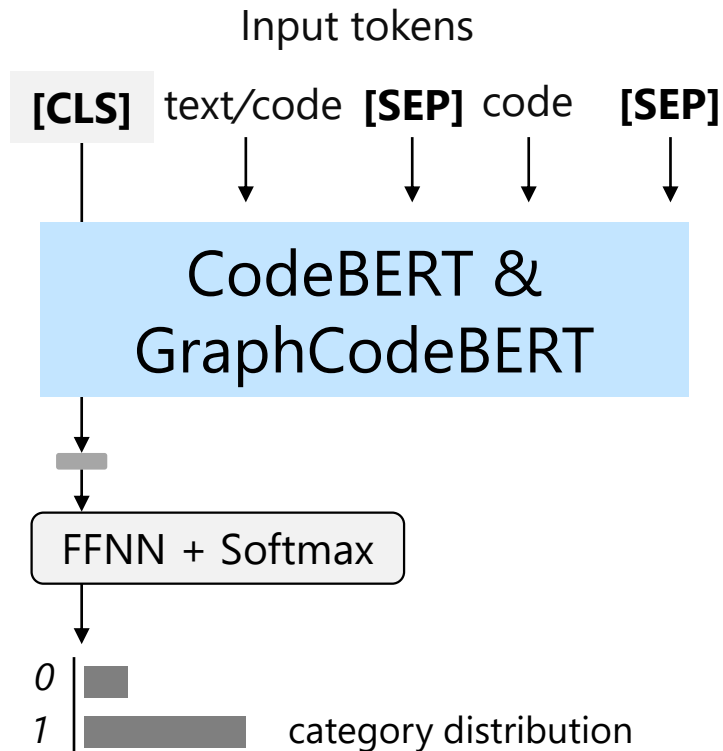
Results on code refinement.

Model	Small		Medium	
	BLEU	Accuracy	BLEU	Accuracy
Naïve	78.06	0.0	90.91	0.0
LSTM	76.76	10.0	72.08	2.5
RoBERTa (code)	77.30	15.9	90.07	4.1
CodeBERT	77.42	16.4	90.07	5.2
GraphCodeBERT	<b>80.02</b>	<b>17.3</b>	<b>91.31</b>	<b>9.1</b>



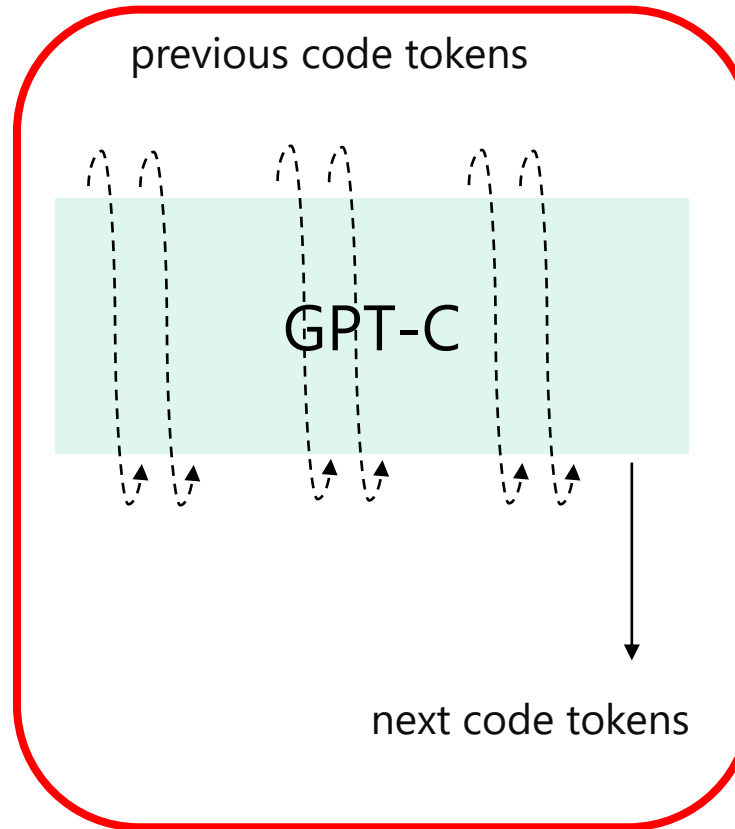
# Three Pre-trained Models for Code

## Understanding



Accepted by EMNLP 2020 and ICLR 2021.

## Generation



Submitted to EMNLP 2021.

Input Code

Grammformer Encoder

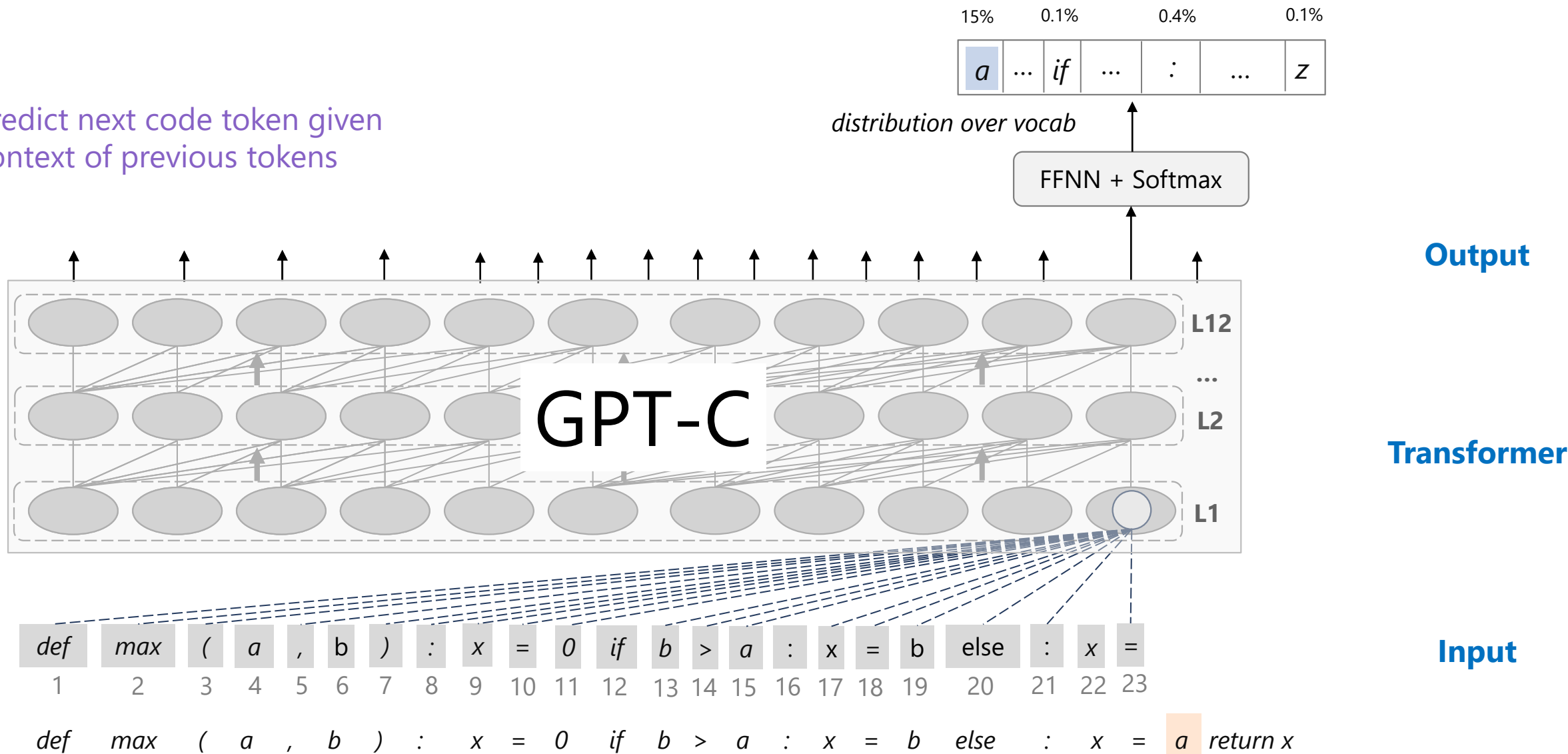
Grammformer Decoder

Output Code Sketch with Holes

Submitted to NeurIPS 2021.

# GPT-C: Multilingual Pre-trained Model

Predict next code token given context of previous tokens



Trained for 10 PLs: JavaScript, C, Java, Go, PHP, Python, C++, C#, Ruby, TypeScript



# Evaluation on Code Completion

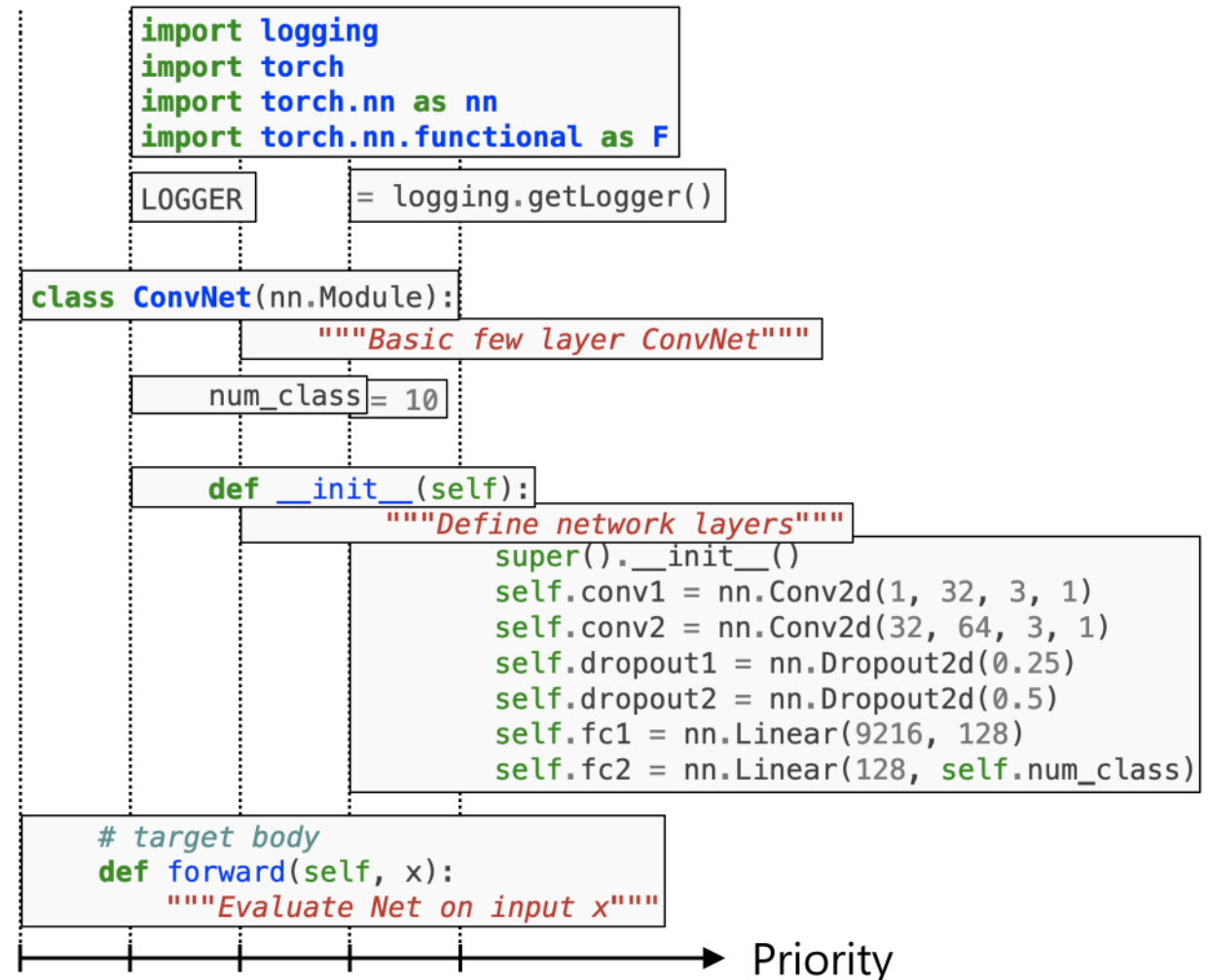
- Corpus
  - 10 programming languages
  - 354K code projects
  - 18B code lines
  - Split 8:1:1 as Train/Dev/Test

Programming language	#Projects	#Files (↓)	#Lines
JavaScript	113,890	15,330,706	4,226,121,235
C	19,900	13,462,890	7,253,471,852
Java	46,921	10,385,540	1,491,132,997
Go	17,922	5,720,219	1,997,845,604
PHP	24,625	4,691,140	653,891,761
Python	71,343	4,465,808	854,503,198
C++	20,958	4,293,413	1,400,309,370
C#	17,387	3,765,835	550,267,681
Ruby	17,804	1,663,262	137,558,948
TypeScript	3,801	466,924	64,671,728

Model	Test language	ROUGE-L			Average edit similarity (Levenshtein, %)
		Precision	Recall	F1	
GPT-C (12L)	Python	0.72	0.80	0.76	83.0
	C#	0.56	0.75	0.64	77.5
	JavaScript	0.71	0.81	0.76	92.2
	Go	0.65	0.72	0.68	81.0
	Scala <b>(zero-shot)</b>	0.41	0.54	0.47	64.2

# Extended Context for Code Completion

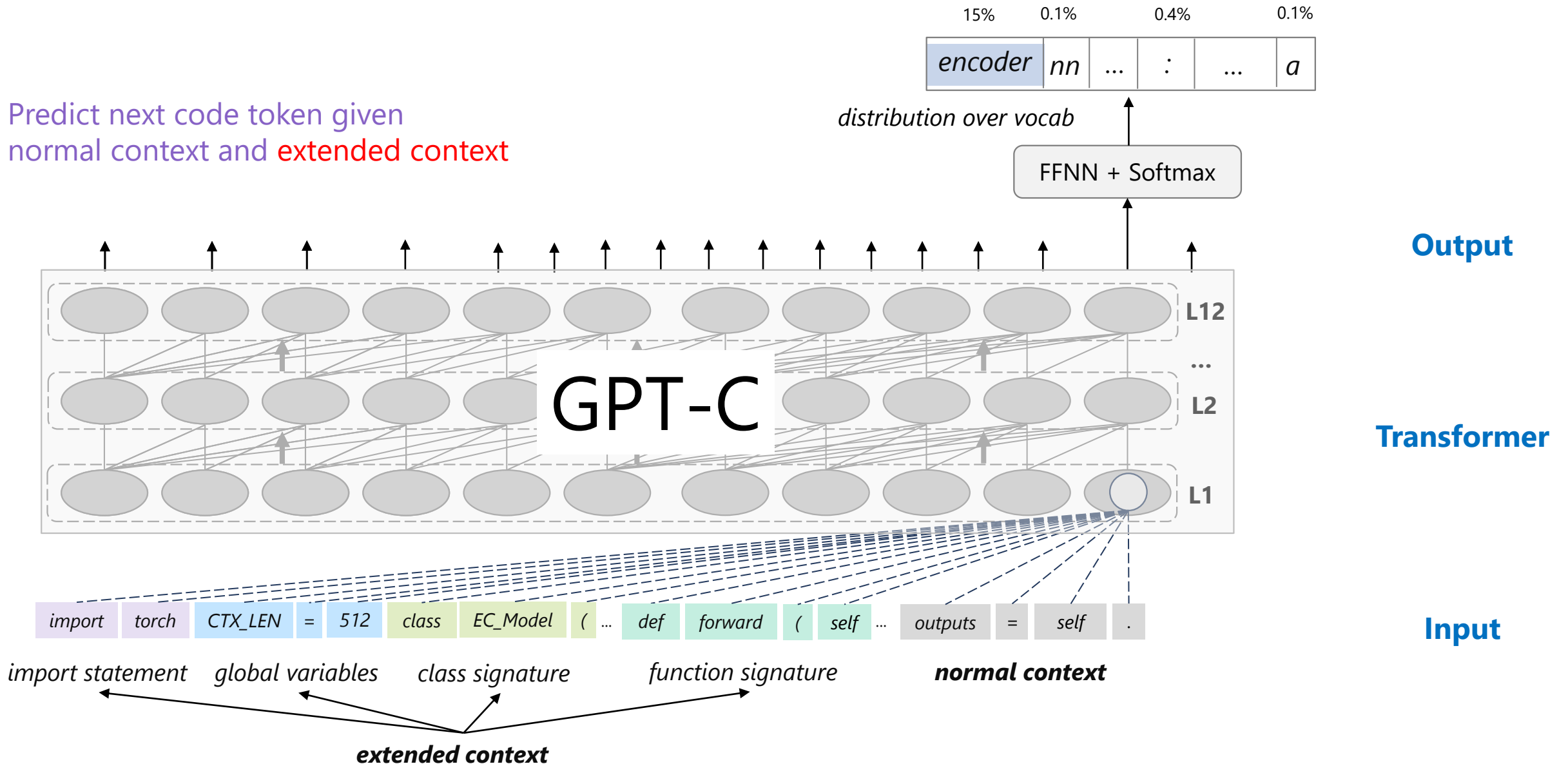
1. Take the concrete syntax tree of the source file;
2. Prioritize the syntactic elements;
  - i. signature and docstring of the focal method;
  - ii. global import statements and assigned values;
  - iii. class attributes, peer class method signatures, class docstring, peer class method docstrings;
  - iv. global expressions and code bodies of peer class methods.
3. Take elements based on their priorities until the context window has been filled.



We reserve 3/4 (768/1,024) tokens for the extended context and 1/4 (256/1024) tokens for the local context.

# GPT-C with Extended Context

Predict next code token given  
normal context and **extended context**



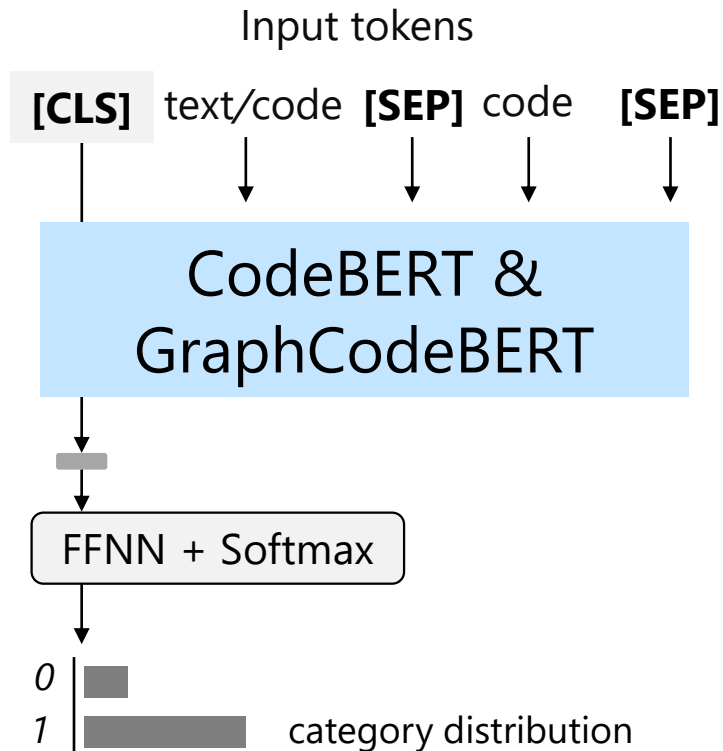
# Evaluation on Code Completion

- Pre-training data
  - Python dataset used in multilingual GPT-C pre-training
- Evaluation data
  - PY150 test set in CodeXGLUE

Model	Test language	ROUGE-L			Average edit similarity (Levenshtein, %)
		Precision	Recall	F1	
GPT-C (12L)	Python	0.81	0.94	0.87	89.0
GPT-C with Extended Context (12L)	Python	<b>0.90</b>	<b>0.96</b>	<b>0.93</b>	<b>93.7</b>

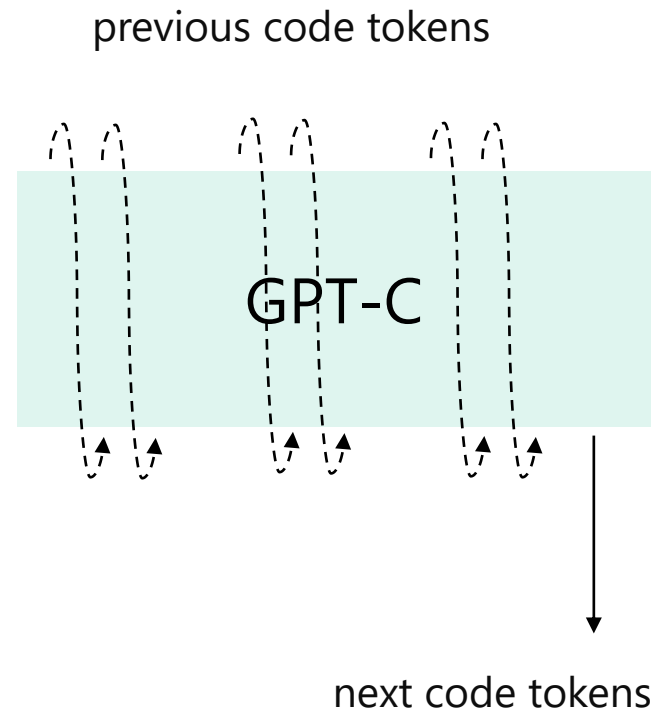
# Three Pre-trained Models for Code

## Understanding

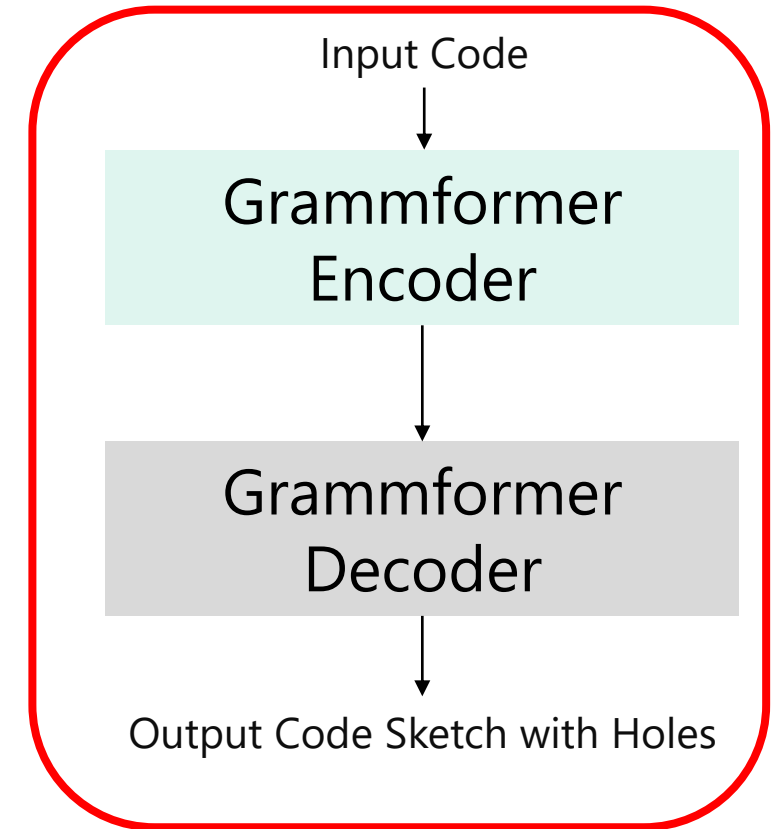


Accepted by EMNLP 2020 and ICLR 2021.

## Generation



Submitted to EMNLP 2021.




Submitted to NeurIPS 2021.

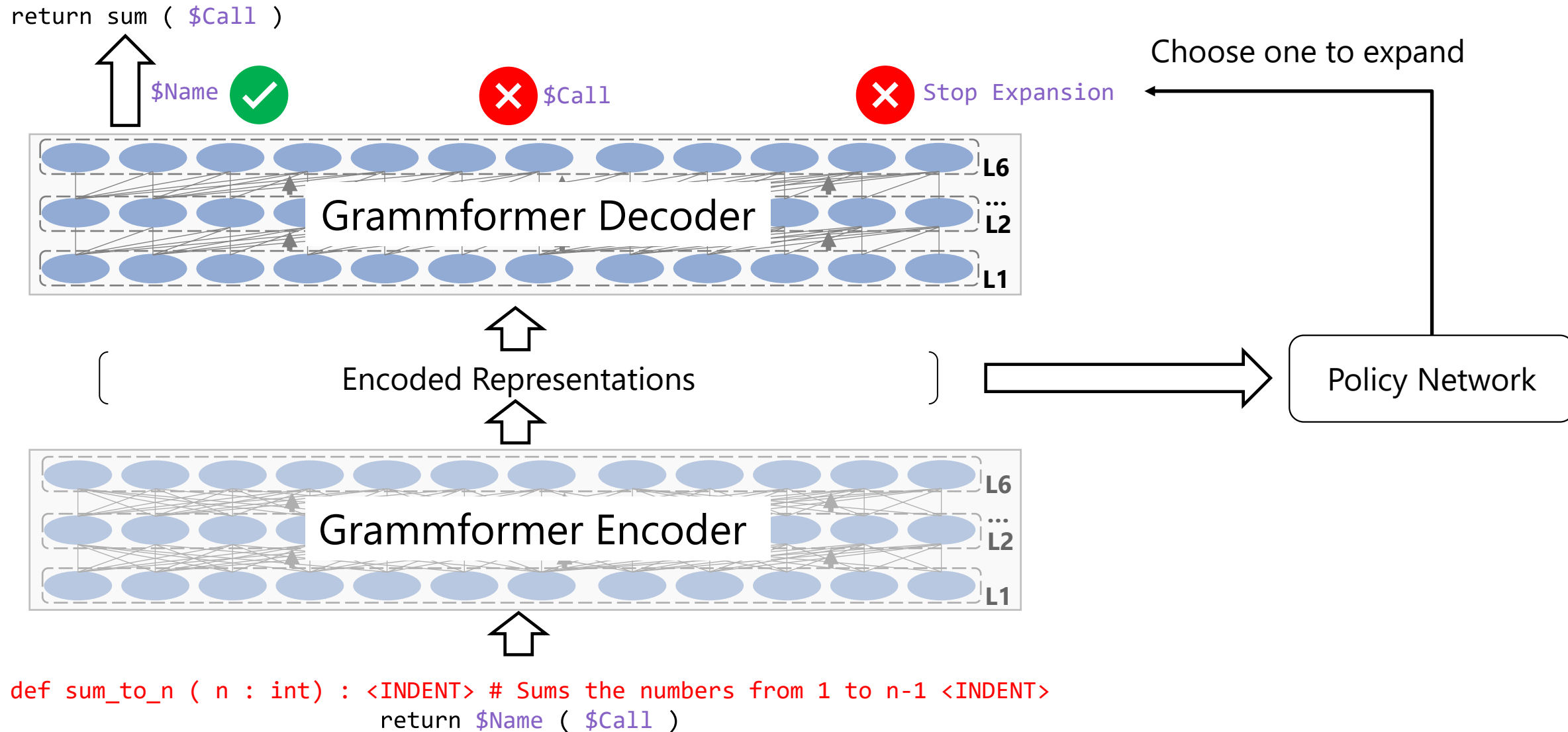
# Challenge in Code Completion: Ambiguities

```
1 import argparse
2 ap = argparse.ArgumentParser()
3 ap.add_argument("--release",
4                 action="store_true")
5 ap.add_argument("--prerelease",
6                 action="store_true")
7 I
```

# Grammformer Example

x <sub>0</sub> :	r	=	$\langle \text{Expr} \rangle$							i <sub>0</sub> = 3		
x <sub>1</sub> :	r	=	$\langle \text{Expr} \rangle$	*	$\langle \text{ParenthesizedExpr} \rangle$					i <sub>1</sub> = 5		
x <sub>2</sub> :	r	=	$\langle \text{Expr} \rangle$	*	(	$\langle \text{Expr} \rangle$				)	i <sub>2</sub> = 6	
x <sub>3</sub> :	r	=	$\langle \text{Expr} \rangle$	*	(	$\langle \text{Expr} \rangle$	-	$\langle \text{Expr} \rangle$		)	i <sub>3</sub> = 8	
x <sub>4</sub> :	r	=	$\langle \text{Expr} \rangle$	*	(	$\langle \text{Expr} \rangle$	-	$\langle \text{Identifier} \rangle$	(	$\langle \text{ArgList} \rangle$	)	i <sub>4</sub> = 8
x <sub>5</sub> :	r	=	$\langle \text{Expr} \rangle$	*	(	$\langle \text{Expr} \rangle$	-	foo	(	$\langle \text{ArgList} \rangle$	)	i <sub>5</sub> = 10
x <sub>6</sub> :	r	=	$\langle \text{Expr} \rangle$	*	(	$\langle \text{Expr} \rangle$	-	foo	(	$\langle \text{Identifier} \rangle$	)	i <sub>6</sub> = 10
x <sub>7</sub> :	r	=	$\langle \text{Expr} \rangle$	*	(	$\langle \text{Expr} \rangle$	-	foo	(	args	)	i <sub>7</sub> = 6
x <sub>8</sub> :	r	=	$\langle \text{Identifier} \rangle$	*	(	$\langle \text{Expr} \rangle$	-	foo	(	args	)	i <sub>8</sub> = 6
x <sub>9</sub> :	r	=	x	*	(	$\langle \text{Expr} \rangle$	-	foo	(	args	)	i <sub>9</sub> = 

# Grammformer Inference





# Grammformer Training

---

**Algorithm 1** GRAMMFORMER generative process, given an input sequence  $\mathbf{x}_0$ .

---

```

for  $t = 0, 1, 2, \dots$  do
     $i_t \sim P_s(i|\mathbf{x}_t, N(\mathbf{x}_t))$   $\triangleright$  Sample non-terminal position from  $N(\mathbf{x}_t)$  to expand
    if  $i_t = \textcircled{\times}$  then  $\triangleright$  if  $\mathbf{x}_t$  does not contains non-terminals or none was selected by  $P_s$ 
        break  $\triangleright$  Stop generation
     $\hat{\mathbf{y}}_{t \odot i_t} \sim P_e(\mathbf{y}|\mathbf{x}_t, i_t)$   $\triangleright$  Sample expansion of non-terminal at position  $i_t$ 
     $\mathbf{x}_{t+1} \leftarrow \mathbf{x}_{t, < i_t} \mathbin{::} \hat{\mathbf{y}}_{t \odot i_t} \mathbin{::} \mathbf{x}_{t, > i_t}$   $\triangleright$  Create  $\mathbf{x}_{t+1}$  by expanding non-terminal at  $i_t$  to  $\hat{\mathbf{y}}_{t \odot i_t}$ 
 $\mathbf{x}_{\text{out}} \leftarrow \text{NONTERMINALSTOHOLES}(\mathbf{x}_t)$   $\triangleright$  Convert any remaining non-terminals to holes
return  $\mathbf{x}_{\text{out}}$ 

```

---

$$\mathcal{L}_{\text{train}}(\mathbf{x}_0, \mathbf{x}^*) = (r(\mathbf{x}_{\text{out}}, \mathbf{x}^*) - \tilde{r}(\mathbf{x}_0)) \sum_{t=0}^T (-\log P_s(i_t|\mathbf{x}_t) - \mathbb{I}(i_t \neq \textcircled{\times}) \log P_e(\hat{\mathbf{y}}_{t \odot i_t}|\mathbf{x}_t, i_t))$$

# REGEXACC as Reward Function

$$\text{REGEXACC}(\hat{s}, s^*) \triangleq \frac{\text{nTerm}(\hat{s})}{\text{nTerm}(s^*)} \cdot \text{matches}(\text{toRegex}(\hat{s}), s^*)$$

return the number of terminal symbols

return 1 if the regex matches the ground truth, and 0 otherwise

turn a predicted sketch into a regular expression by replacing all non-terminals with the wildcard matching any non-empty sequence

# Experiment

## Dataset

More than 20 stars in GitHub that contain C# and Python Code.  
Including 3.8M and 4.5 M files for C# and Python, respectively.

## Results

	C#				Python			
	REGEXACC		ROUGE	Avg Gen Length	REGEXACC		ROUGE	Avg Gen Length
	Top 1	Top 5			Top 1	Top 5		
$L \rightarrow R$	0.42	0.47	77.0	7.1	0.17	0.20	<b>53.2</b>	5.8
$L \rightarrow R + \text{⊗}$	0.45	0.54	69.1	5.3	0.20	0.29	39.3	3.0
GRAMMFORMER	<b>0.47</b>	<b>0.59</b>	<b>77.4</b>	<b>7.5</b>	<b>0.21</b>	<b>0.30</b>	51.6	<b>6.1</b>

# Today's Agenda

- Background
- Pre-trained Models for Code Intelligence
- **Benchmark for Code Intelligence**
- Conclusion & Future Work

# CodeXGLUE: 14 datasets for 10 Code-related tasks

Category	Task	Dataset Name	Language	Train/Dev/Test Size	Baselines	Dataset Provider	Task definition
Code-Code	Clone Detection	BigCloneBench	Java	900K/416K/416K	CodeBERT	<a href="#">Univ. of Saskatchewan</a>	Predict semantic equivalence for a pair of codes.
		POJ-104	C/C++	32K/8K/12K		<a href="#">Peking Univ</a>	Retrieve semantically similar codes.
	Defect Detection	Defects4J	C	21k/2.7k/2.7k		<a href="#">Univ. of Washington</a>	Identify whether a function is vulnerable.
	Cloze Testing	CT-all	Python, Java, PHP, JavaScript, Ruby, Go	-/-/176k		<a href="#">Created by MSRA based on CodeSearchNet</a>	Tokens to be predicted come from the entire vocab.
		CT-max/min	Python, Java, PHP, JavaScript, Ruby, Go	-/-/2.6k		<a href="#">Created by MSRA based on CodeSearchNet</a>	Tokens to be predicted come from {max, min}.
	Code Completion	PY150	Python	100k/5k/50k	CodeGPT	<a href="#">ETH Zurich, line-level data added by MSRA</a>	Predict following tokens given contexts of codes.
		GitHub Java Corpus	Java	13k/7k/8k		<a href="#">Univ. of Edinburgh, line-level data added by MSRA</a>	
	Code Refinement	Bugs2Fix	Java	98K/12K/12K	Encoder-Decoder	<a href="#">The College of William and Mary</a>	Automatically refine codes by fixing bugs.
	Code Translation	CodeTrans	Java-C#	10K/0.5K/1K		MSRA	Translate the codes from one programming language to another programming language.
Text-Code	NL Code Search	CodeSearchnet, AdvTest	Python	251K/9.6K/19K	CodeBERT	<a href="#">GitHub + MSR Cambridge, test provided by MSRA</a>	Given a natural language query as input, find semantically similar codes.
		StacQC, WebQueryTest	Python	2.9k/0.9k/1.9k		<a href="#">The Ohio State Univ, test provided by MSRA</a>	Given a pair of natural language and code, predict whether they are relevant or not.
	Text-to-Code Generation	CONCODE	Java	100K/2K/2K	CodeGPT	<a href="#">Univ. of Washington</a>	Given a natural language docstring/comment as input, generate a code.
Code-Text	Code Summarization	CodeSearchNet*	Python, Java, PHP, JavaScript, Ruby, Go	908K/45K/53K	Encoder-Decoder	<a href="#">Filtered based on CodeSearchNet data</a>	Given a code, generate its natural language docstring/comment.
Text-Text	Documentation Translation	Microsoft Docs	English-Latvian/Danish/Norwegian/Chinese	156K/4K/4K		MSRA	Translate code documentation between human languages (e.g. En-Zh), intended to test low-resource multi-lingual translation.

# CodeXGLUE: 14 datasets for 10 Code-related tasks

Search or jump to... Pull requests Issues Marketplace Explore

microsoft / CodeXGLUE Watch 23 Star 344 Fork 95

<> Code Issues Pull requests Actions Projects Wiki Security Insights

main 2 branches 0 tags

guody5 Merge pull request #55 from ncoop57/... 8bf55ca 11 days ago 446 commits

- Code-Code Update dataflow\_match.py last month
- Code-Text/code-to-text Added scripts to run in Google Colab 2 months ago
- Text-Code Add reference to the finetuned codebert model 11 days ago
- Text-Text/text-to-text Update run-multi.sh 8 months ago
- webpage\_files Update code2text\_generation.json 11 days ago
- .gitignore Initial commit 9 months ago
- CODE\_OF\_CONDUCT.md Initial CODE\_OF\_CONDUCT.md commit 9 months ago
- Data\_LICENCE Update Data\_LICENCE 9 months ago
- LICENSE Initial LICENSE commit 9 months ago
- README.md Update README.md 3 months ago
- SECURITY.md Initial SECURITY.md commit 9 months ago
- baselines.jpg Add files via upload 7 months ago
- index.html add url function to webpage last month
- tasks.jpg Add files via upload 7 months ago
- time-cost.jpg Add files via upload 7 months ago

README.md

## Introduction

According to [Evans Data Corporation](#), there are 23.9 million professional developers in 2019, and the population is expected to reach 28.7 million in 2024. With the growing

<https://github.com/microsoft/CodeXGLUE>

CodeXGLUE Home Intro Leaderboard Submission

**Microsoft**

CodeXGLUE stands for General Language Understanding Evaluation benchmark for CODE. It includes 14 datasets for 10 diversified programming language tasks covering code-code (clone detection, defect detection, cloze test, code completion, code refinement, and code-to-code translation), text-code (natural language code search, text-to-code generation), code-text (code summarization) and text-text (documentation translation) scenarios. We provide three baseline models to support these tasks, including BERT-style pre-trained model (i.e. [CodeBERT](#)) which is good at understanding problems, GPT-style pre-trained model which we call [CodeGPT](#) to support completion and generation problems, and Encoder-Decoder framework that supports sequence-to-sequence generation problems.

## Overall Leaderboard

Rank	Model	Organization	Date	clone detection	defect detections...	cloze test
1	<a href="#">CodeBERT Baseline</a>	CodeXGLUE Team	2020-08-30	90.40	62.08	84.78

## Clone Detection (Code-Code)

Rank	Model	Organization	Date	Precision	Recall	F1
1	<a href="#">PLBART</a>	PLBART(UCLA, ...)	2021-04-02	/	/	0.972
2	<a href="#">CodeBERT</a>	CodeXGLUE Team	2020-08-30	0.960	0.969	0.965
3	<a href="#">RoBERTa</a>	CodeXGLUE Team	2020-08-30	0.935	0.965	0.949

<https://microsoft.github.io/CodeXGLUE/>

# Current Submission Status

	Task	Dataset	# of Submissions	Organizations
Code-Code	Clone Detection	BigCloneBench	2	UCLA & Columbia University; HNUST;
		POJ-104	2	UCLA & Columbia University;
	Defect Detection	Defects4J	4	IBM Research; Case Western Reserve University; UCLA & Columbia University; INESC-ID & Carnegie Mellon University;
	Cloze Test	CT-all		
		CT-max/min		
	Code Completion	GitHub Java Corpus		
		PY150		
	Code Repair	Bugs2Fix	2	Case Western Reserve University; UCLA & Columbia University;
	Code Translation	CodeTrans	1	UCLA & Columbia University;
Text-Code	NL Code Search	AdvTest		
		WebQueryTest		
	Text-to-code Generation	CONCODE	5	Wuhan University; Case Western Reserve University; UCLA & Columbia University; UBC Research;
Code-Text	Code Summarization	CodeSearchNet	4	USTC & MSRA; Case Western Reserve University; UCLA & Columbia University; UC Davis;
Text-Text	Documentation Translation	Microsoft Docs		
			20 (in total)	

# Today's Agenda

- Background
- Pre-trained Models for Code Intelligence
- Benchmark for Code Intelligence
- **Conclusion & Future Work**



# Conclusion & Future Work

- Conclusion
  - Large-scale self-supervised pre-training can be successfully adapted to code scenarios.
  - Considering the uniqueness of code can lead to better pre-trained models for code.
  - CodeXGLUE is a useful benchmark and model evaluation platform for code intelligence.
- Future work
  - Pre-trained models with more code-related knowledge (syntax, structure, etc.) integrated
  - Pre-trained models to deal with downstream tasks with very long code inputs/outputs
  - Efficient training and inference for code completion/generation tasks
  - CodeXGLUE++ with more programming languages and more tasks
  - Serious consideration on privacy and intellectual property when using open-source codes in pre-training and downstream applications (e.g., OpenAI's Copilot 😊)

**Thank You**  
**谢谢**