CNCC

多维度编程语言预训练及实际应用

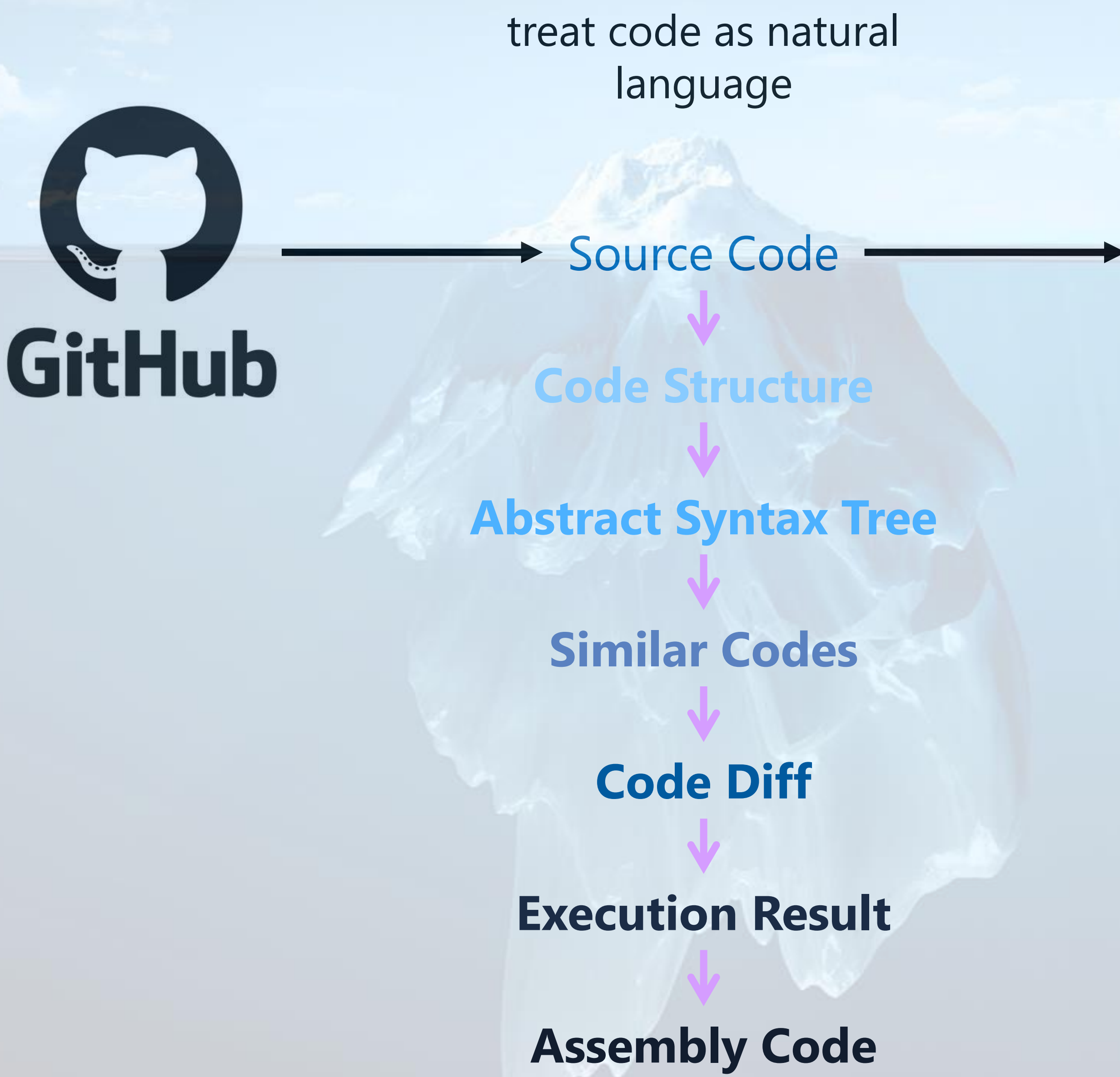
Nan Duan (段楠)

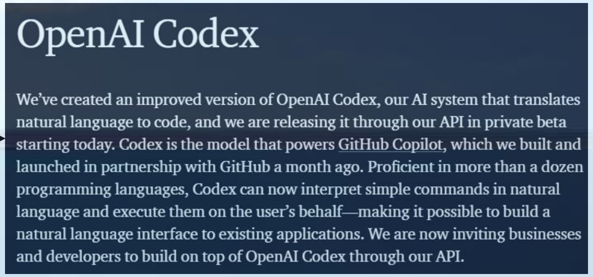
Senior Principal Research Manager

Microsoft Research Asia

2022-12-10 @ CNCC 2022

Large-scale Pre-trained Models for Code Intelligence

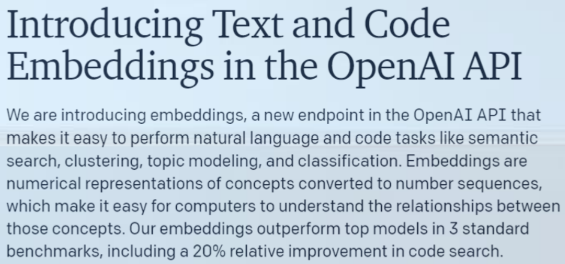




OpenAI Codex

We've created an improved version of OpenAI Codex, our AI system that translates natural language to code, and we are releasing it through our API in private beta starting today. Codex is the model that powers GitHub Copilot, which we built and launched in partnership with GitHub a month ago. Proficient in more than a dozen programming languages, Codex can now interpret simple commands in natural language and execute them on the user's behalf—making it possible to build a natural language interface to existing applications. We are now inviting businesses and developers to build on top of OpenAI Codex through our API.

Codex (OpenAI)



Introducing Text and Code Embeddings in the OpenAI API

We are introducing embeddings, a new endpoint in the OpenAI API that makes it easy to perform natural language and code tasks like semantic search, clustering, topic modeling, and classification. Embeddings are numerical representations of concepts converted to number sequences, which make it easy for computers to understand the relationships between those concepts. Our embeddings outperform top models in 3 standard benchmarks, including a 20% relative improvement in code search.

Embeddings (OpenAI)

(Some of) Our Work @ MSRA

Code Representation

- **CodeBERT** (EMNLP 2020): propose the 1st code-text pre-trained model
- **GraphCodeBERT** (ICLR 2021): use variable relationship to enhance code representation
- **UniXcoder** (ACL 2022): use sequential AST to enhance code representation

Code Retrieval

- **AR2** (ICLR 2022): propose a general dense retrieval framework that trains retriever & ranker in a minimax adversarial manner and can cover text-to-text, text-to-code, and code-to-code tasks
- **CodeRetriever** (EMNLP 2022): extract code-text and code-code pairs from Web for building code retrieval models with unimodal and bimodal contrastive learning

Code Generation

- **GPT-C w/ Extended Context** (EMNLP 2021): improve GPT-C for code completion with extended context
- **Grammformer** (ICLR 2022): generate codes based on code grammar and predict holes to alleviate the uncertainty issue
- **ReACC** (ACL 2022): propose a retrieval-augmented code completion framework, which uses the partial code as a query to retrieve similar codes and predicts the following codes based on the retrieved similar codes

Code Review & Refinement

- **CodeReviewer** (ESEC/FSE 2022): propose an encoder-decoder based pre-trained model for automating code review activities, such as code diff quality estimation, review generation, code refinement based on code review, etc.
- **CodeReviewer v2.0** (on-going 2022): propose a retrieval-augmented model for automating code review activities, like ReACC for code completion.

Code Benchmark

- **CodeXGLUE** (NeurIPS 2021): build a benchmark for code intelligence, which covers 14 datasets for 10 code tasks
- **CoSQA** (ACL 2021): build a benchmark for text-to-code search based on Web data
- **CodeExplanation** (EMNLP 2022): build a benchmark for code-to-explanation generation

Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, Ming Zhou. **CodeBERT: A Pre-Trained Model for Programming and Natural Language**. EMNLP 2020.

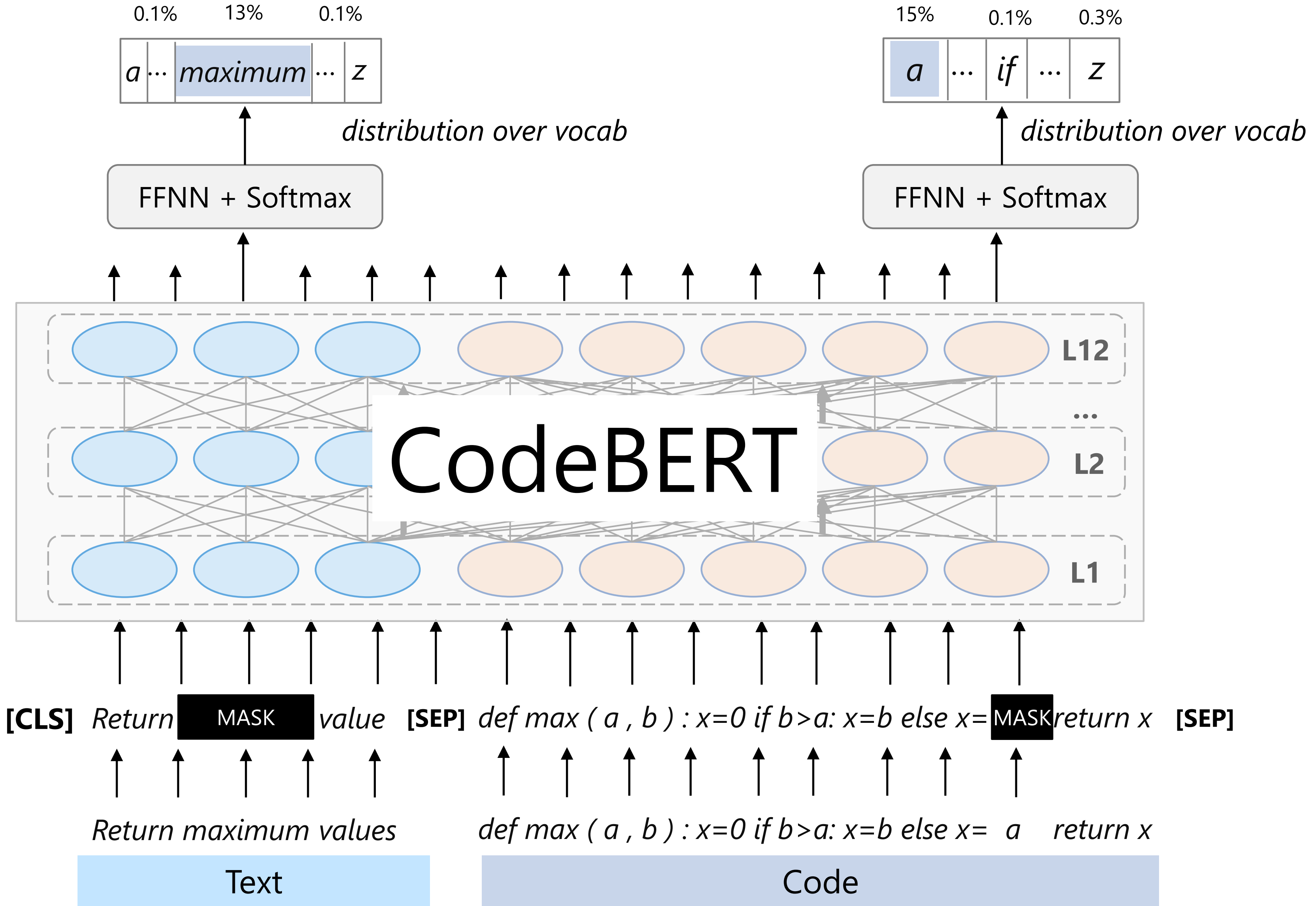
CodeBERT (v1): Pre-Train with Code+Text

Source code

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

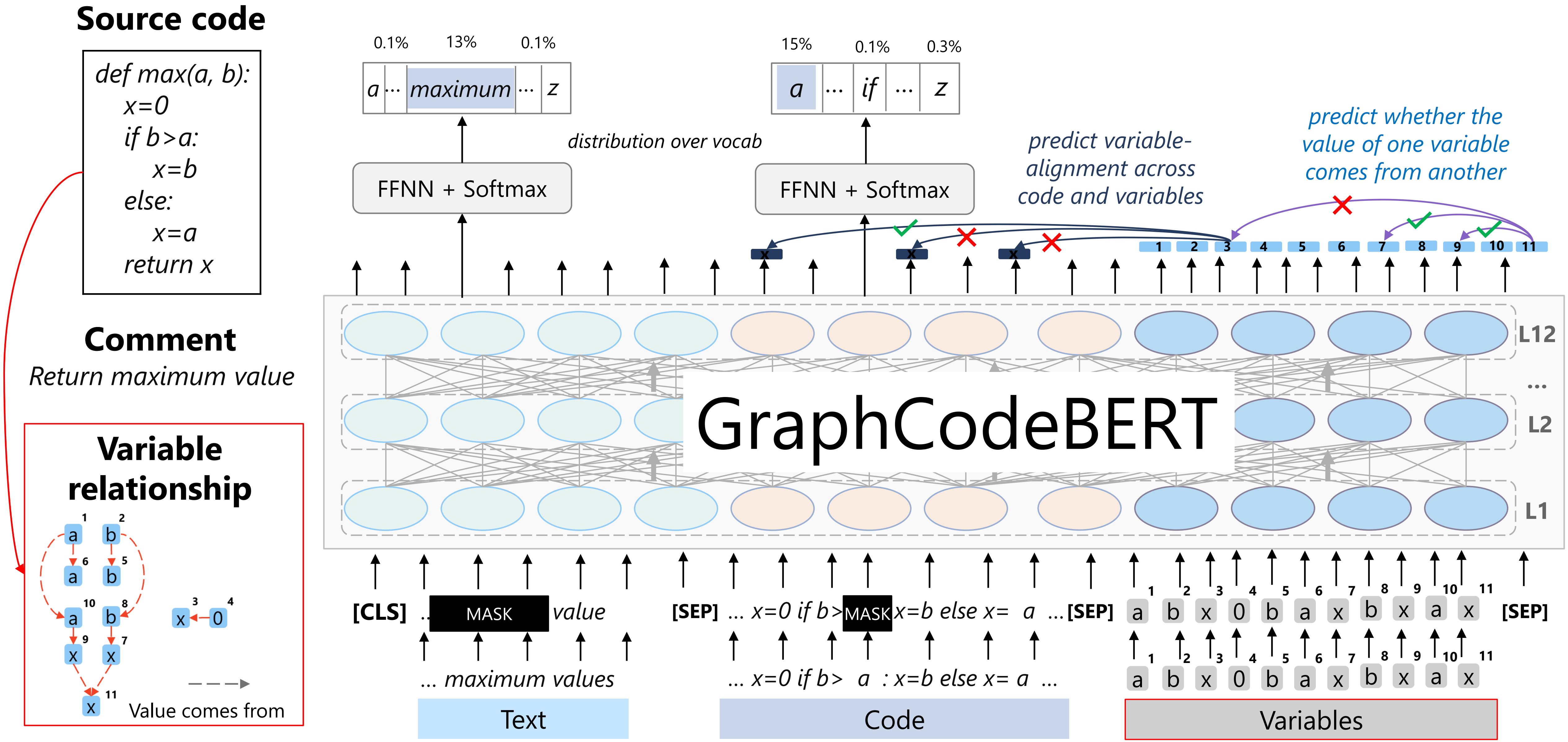
Comment

Return maximum value



Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Alexey Svyatkovskiy, Shengyu Fu, Michele Tufano, Shao Kun Deng, Colin Clement, Dawn Drain, Neel Sundaresan, Jian Yin, Daxin Jiang, Ming Zhou. **GraphCodeBERT: Pre-training Code Representations with Data Flow**. ICLR 2021.

GraphCodeBERT (v2): Pre-Train with Code+Text+VarRel



Source code

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

Comment

Return maximum value

Variable relationship

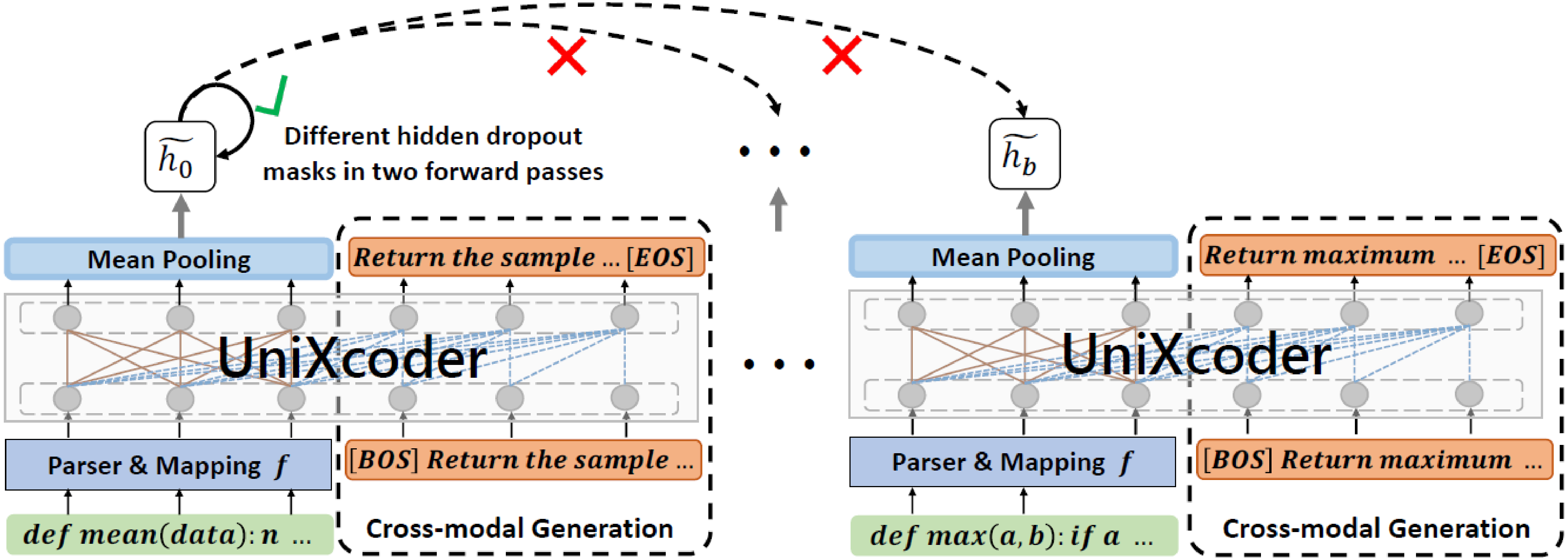
Value comes from

distribution over vocab

predict variable-alignment across code and variables

predict whether the value of one variable comes from another

UniXcoder (v3): Pre-Train with Code+Text+SeqAST



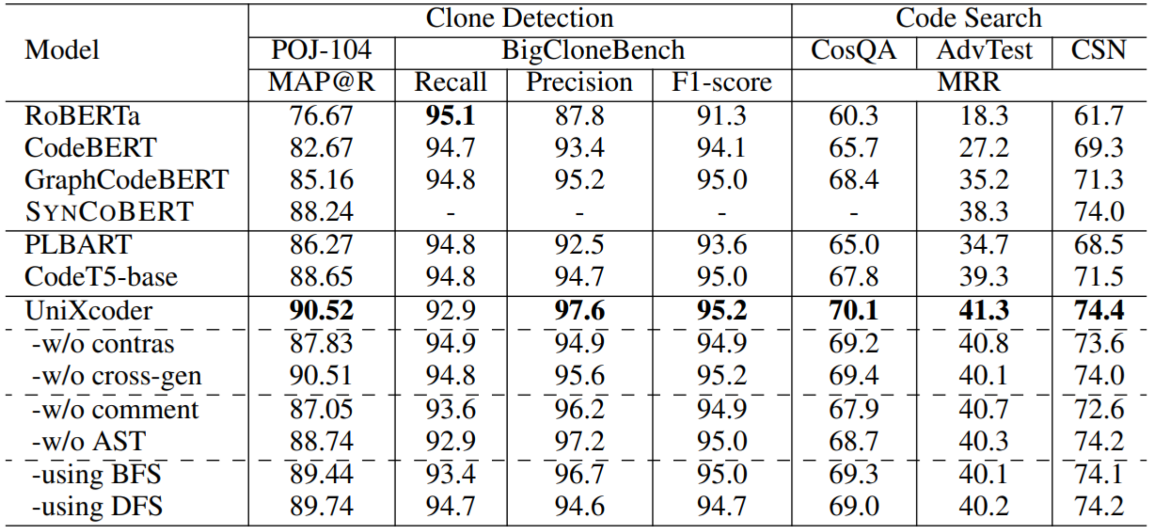
$\widetilde{h}_0$
Different hidden dropout masks in two forward passes
$\widetilde{h}_b$
Mean Pooling
UniXcoder
Parser & Mapping f
$def\ mean(data): n$ ...
$Return\ the\ sample$ ... $[EOS]$
$[BOS]\ Return\ the\ sample$ ...
Cross-modal Generation
Mean Pooling
UniXcoder
Parser & Mapping f
$def\ max(a, b): if\ a$ ...
$Return\ maximum$ ... $[EOS]$
$[BOS]\ Return\ maximum$ ...
Cross-modal Generation

- **AST-based code-to-code retrieval** forwards the same sequential AST input using a different hidden dropout mask as a positive example and uses other sequential ASTs in the same batch as negative examples.

- **AST-based code-to-text generation** asks the model to generate the comment based on the sequential AST input.

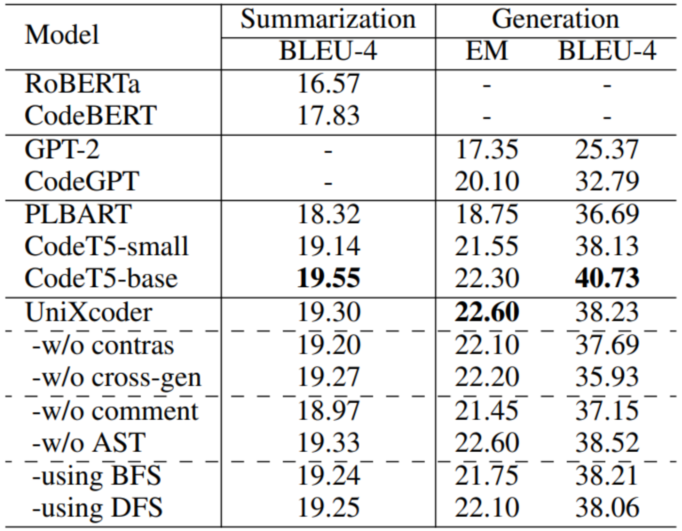
Daya Guo, Shuai Lu, Nan Duan, Yanlin Wang, Ming Zhou, Jian Yin. **UniXcoder: Unified Cross-Modal Pre-training for Code Representation**. ACL 2022.

Evaluation



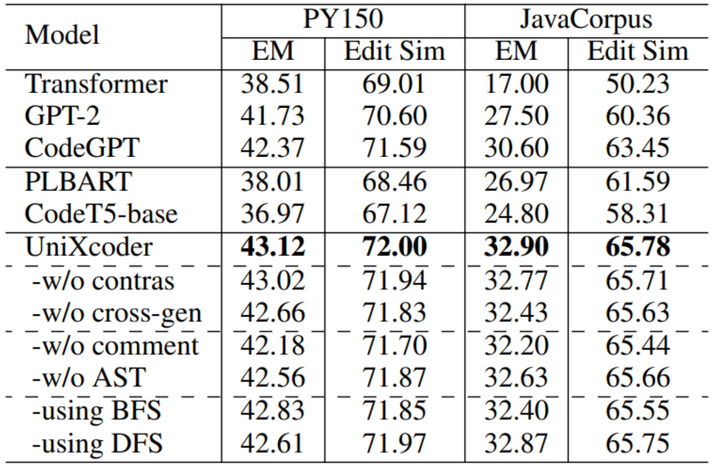
Model	Clone Detection				Code Search		
	POJ-104	BigCloneBench			CosQA	AdvTest	CSN
	MAP@R	Recall	Precision	F1-score	MRR		
RoBERTa	76.67	**95.1**	87.8	91.3	60.3	18.3	61.7
CodeBERT	82.67	94.7	93.4	94.1	65.7	27.2	69.3
GraphCodeBERT	85.16	94.8	95.2	95.0	68.4	35.2	71.3
SynCoBERT	88.24	-	-	-	-	38.3	74.0
PLBART	86.27	94.8	92.5	93.6	65.0	34.7	68.5
CodeT5-base	88.65	94.8	94.7	95.0	67.8	39.3	71.5
UniXcoder	**90.52**	92.9	**97.6**	**95.2**	**70.1**	**41.3**	**74.4**
-w/o contras	87.83	94.9	94.9	94.9	69.2	40.8	73.6
-w/o cross-gen	90.51	94.8	95.6	95.2	69.4	40.1	74.0
-w/o comment	87.05	93.6	96.2	94.9	67.9	40.7	72.6
-w/o AST	88.74	92.9	97.2	95.0	68.7	40.3	74.2
-using BFS	89.44	93.4	96.7	95.0	69.3	40.1	74.1
-using DFS	89.74	94.7	94.6	94.7	69.0	40.2	74.2

Results of code understanding tasks.



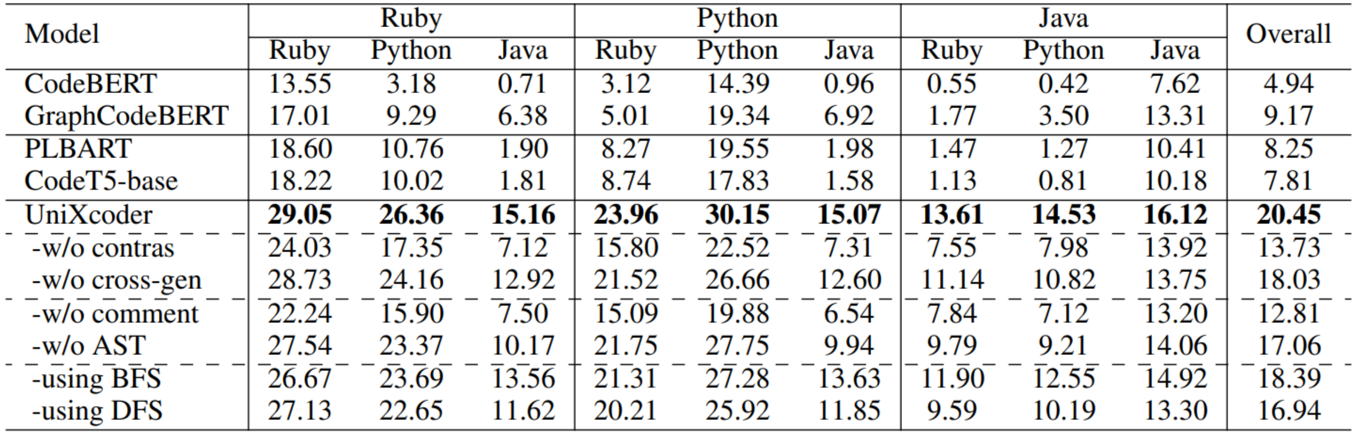
Model	Summarization	Generation	
	BLEU-4	EM	BLEU-4
RoBERTa	16.57	-	-
CodeBERT	17.83	-	-
GPT-2	-	17.35	25.37
CodeGPT	-	20.10	32.79
PLBART	18.32	18.75	36.69
CodeT5-small	19.14	21.55	38.13
CodeT5-base	**19.55**	22.30	**40.73**
UniXcoder	19.30	**22.60**	38.23
-w/o contras	19.20	22.10	37.69
-w/o cross-gen	19.27	22.20	35.93
-w/o comment	18.97	21.45	37.15
-w/o AST	19.33	22.60	38.52
-using BFS	19.24	21.75	38.21
-using DFS	19.25	22.10	38.06

Results of code generation tasks.



Model	PY150		JavaCorpus	
	EM	Edit Sim	EM	Edit Sim
Transformer	38.51	69.01	17.00	50.23
GPT-2	41.73	70.60	27.50	60.36
CodeGPT	42.37	71.59	30.60	63.45
PLBART	38.01	68.46	26.97	61.59
CodeT5-base	36.97	67.12	24.80	58.31
UniXcoder	**43.12**	**72.00**	**32.90**	**65.78**
-w/o contras	43.02	71.94	32.77	65.71
-w/o cross-gen	42.66	71.83	32.43	65.63
-w/o comment	42.18	71.70	32.20	65.44
-w/o AST	42.56	71.87	32.63	65.66
-using BFS	42.83	71.85	32.40	65.55
-using DFS	42.61	71.97	32.87	65.75

Results on code completion task.

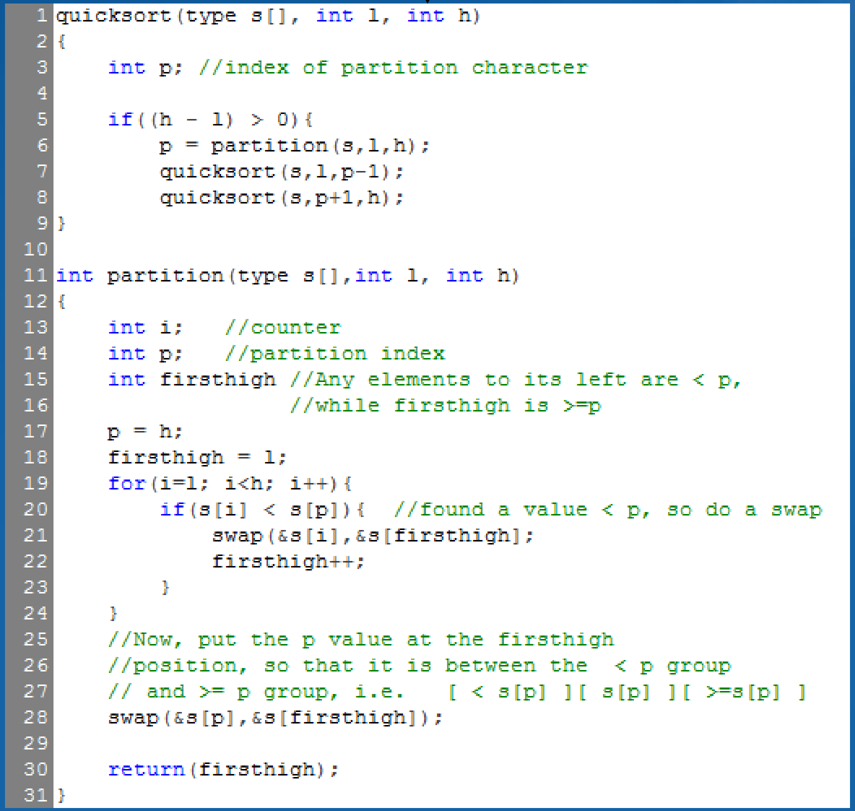


Model	Ruby			Python			Java			Overall
	Ruby	Python	Java	Ruby	Python	Java	Ruby	Python	Java	
CodeBERT	13.55	3.18	0.71	3.12	14.39	0.96	0.55	0.42	7.62	4.94
GraphCodeBERT	17.01	9.29	6.38	5.01	19.34	6.92	1.77	3.50	13.31	9.17
PLBART	18.60	10.76	1.90	8.27	19.55	1.98	1.47	1.27	10.41	8.25
CodeT5-base	18.22	10.02	1.81	8.74	17.83	1.58	1.13	0.81	10.18	7.81
UniXcoder	**29.05**	**26.36**	**15.16**	**23.96**	**30.15**	**15.07**	**13.61**	**14.53**	**16.12**	**20.45**
-w/o contras	24.03	17.35	7.12	15.80	22.52	7.31	7.55	7.98	13.92	13.73
-w/o cross-gen	28.73	24.16	12.92	21.52	26.66	12.60	11.14	10.82	13.75	18.03
-w/o comment	22.24	15.90	7.50	15.09	19.88	6.54	7.84	7.12	13.20	12.81
-w/o AST	27.54	23.37	10.17	21.75	27.75	9.94	9.79	9.21	14.06	17.06
-using BFS	26.67	23.69	13.56	21.31	27.28	13.63	11.90	12.55	14.92	18.39
-using DFS	27.13	22.65	11.62	20.21	25.92	11.85	9.59	10.19	13.30	16.94

Results on zero-shot code-to-code search task.

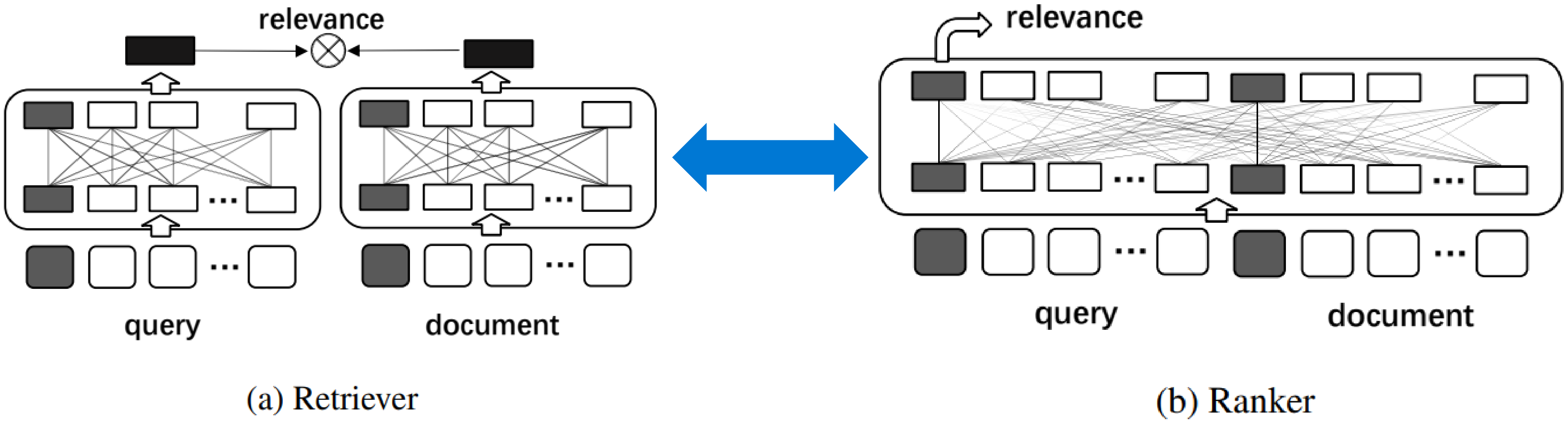
Code Retrieval

I want to write a quick sort algorithm.



```
quicksort(type s[], int l, int h)
{
    int p; //index of partition character

    if((h - l) > 0){
        p = partition(s,l,h);
        quicksort(s,l,p-1);
        quicksort(s,p+1,h);
}

int partition(type s[],int l, int h)
{
    int i;   //counter
    int p;   //partition index
    int firsthigh //Any elements to its left are < p,
                  //while firsthigh is >=p
    p = h;
    firsthigh = l;
    for(i=l; i<h; i++){
        if(s[i] < s[p]){  //found a value < p, so do a swap
            swap(&s[i],&s[firsthigh];
            firsthigh++;
        }
    }
    //Now, put the p value at the firsthigh
    //position, so that it is between the  < p group
    // and >= p group, i.e.   [ < s[p] ][ s[p] ][ >=s[p] ]
    swap(&s[p],&s[firsthigh]);

    return(firsthigh);
}
```

Jointly Optimize Single-tower and Two-tower Models



(a) Retriever

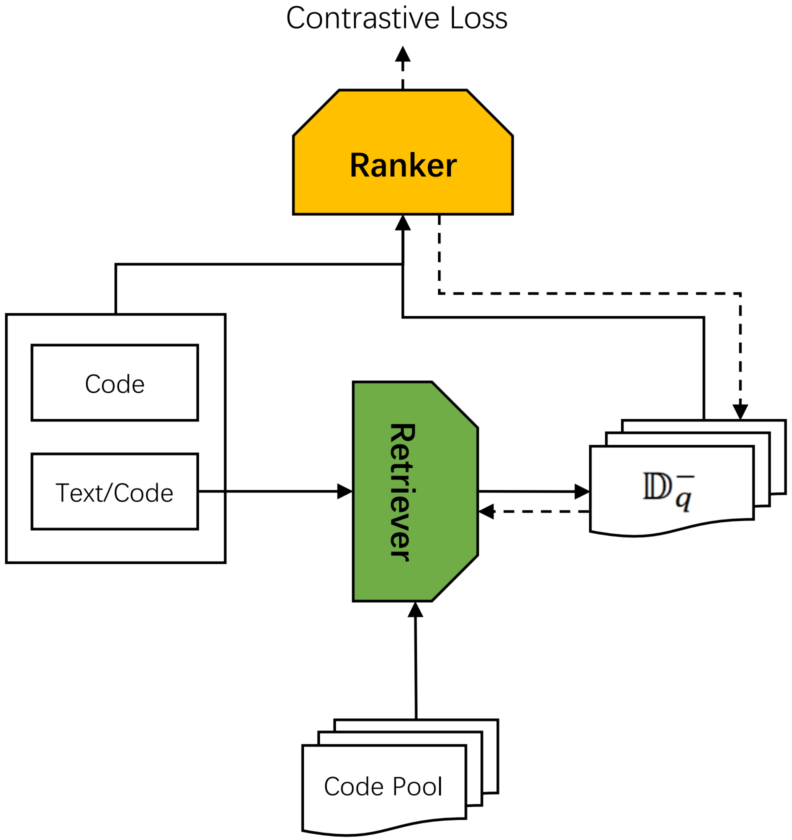
(b) Ranker

Jointly optimize two modules according to a minimax adversarial objective

- Retriever: retrieve negative documents to cheat Ranker
- Ranker: distinguish the ground-truth document and the retrieved ones by Retriever

Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, Weizhu Chen. **Adversarial Retriever-Ranker for Dense Text Retrieval**. ICLR 2022.

Dense Retrieval w/ Adversarial Retriever-Ranker (AR2)



$$J^{G^*,D^*} = \min_\theta \max_\phi \mathbf{E}_{\mathbb{D}_q^- \sim G_\theta(q,\cdot)} \left[\log p_\phi(d|q,d,\mathbb{D}_q^-)\right]$$

$$p_\phi(d|q,d,\mathbb{D}_q^-) = \frac{e^{\tau D_\phi(q,d)}}{e^{\tau D_\phi(q,d)} + \sum_{i=1}^{n} e^{\tau D_\phi(q,d_i^-)}}$$

Retriever θ: try to find the hard negatives $\mathbb{D}_q^-$ to cheat Ranker ϕ.

$$\theta^* = \text{argmin}_\theta J^\theta = \mathbf{E}_{\mathbb{D}_q^- \sim G_\theta(q,\cdot)} \left[\log p_\phi(d|q,d,\mathbb{D}_q^-)\right]$$

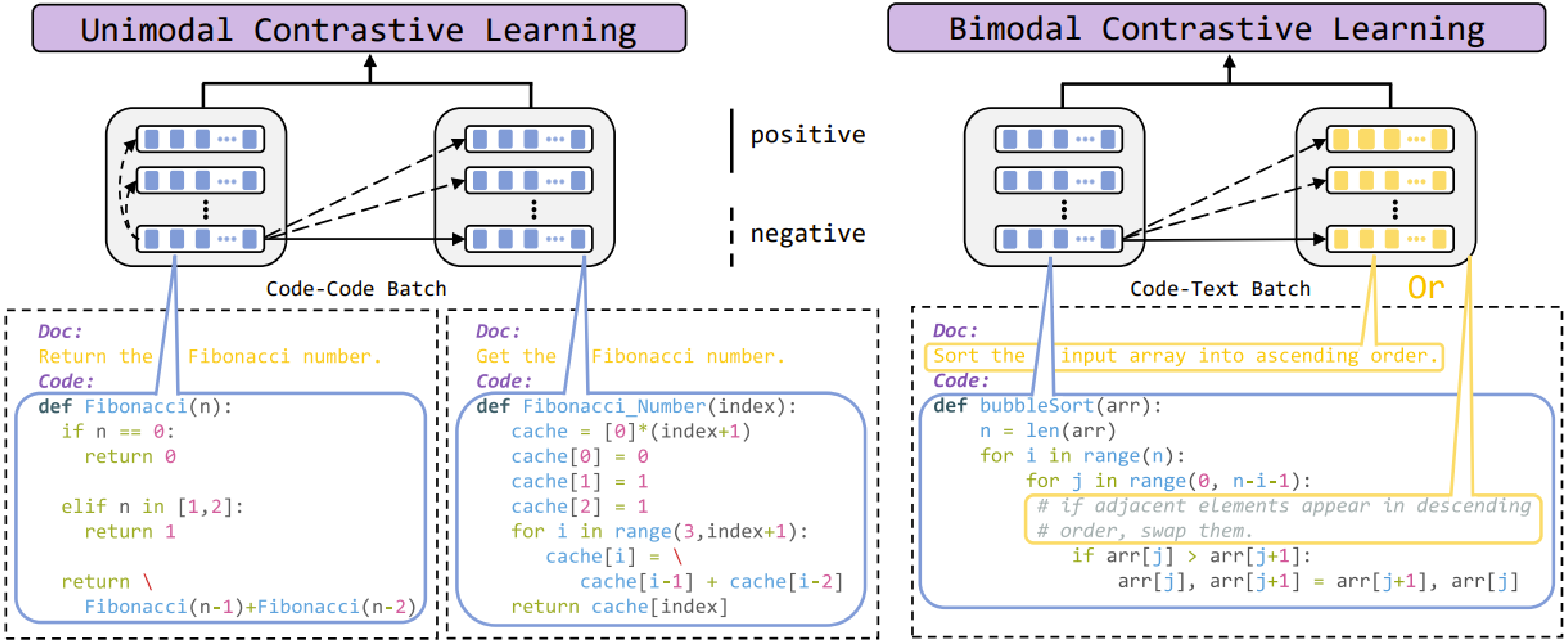
Ranker ϕ: try to find the golden d from the negatives selected by Retriever θ.

$$\phi^* = \text{argmax}_\phi \log p_\phi(d|q,d,\mathbb{D}_q^-)$$

- **Text-Code pairs** come from CodeSearchNet
- **Code-Code pairs** come form AST-based ICT

Hang Zhang, Yeyun Gong, Yelong Shen, Jiancheng Lv, Nan Duan, Weizhu Chen. **Adversarial Retriever-Ranker for dense text retrieval**. ICLR 2022.

CodeRetriever (v1) with Contrastive Learning



```
Doc:
Return the Fibonacci number.
Code:
def Fibonacci(n):
  if n == 0:
    return 0

  elif n in [1,2]:
    return 1

  return \
    Fibonacci(n-1)+Fibonacci(n-2)
```

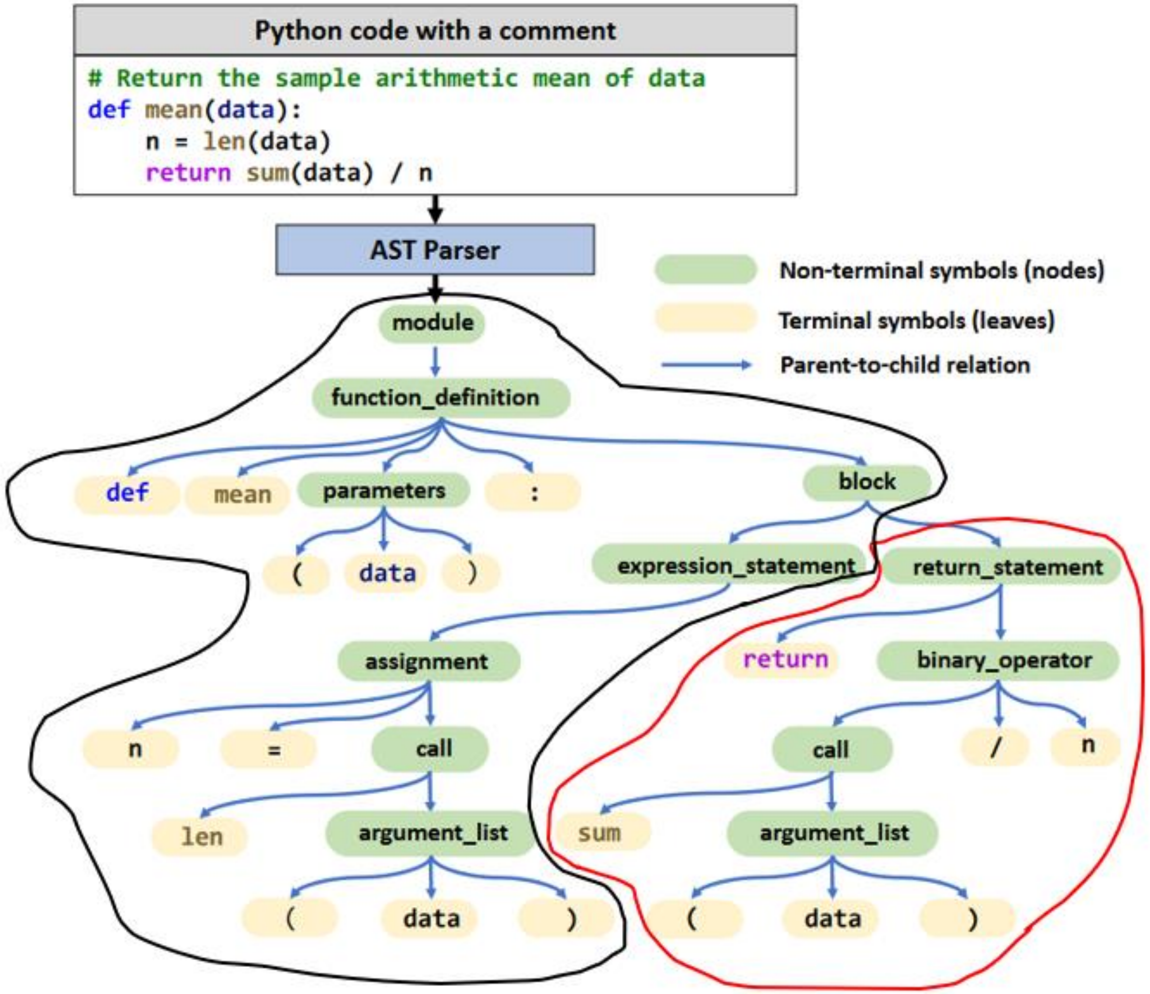
```
Doc:
Get the Fibonacci number.
Code:
def Fibonacci_Number(index):
    cache = [0]*(index+1)
    cache[0] = 0
    cache[1] = 1
    cache[2] = 1
    for i in range(3,index+1):
        cache[i] = \
            cache[i-1] + cache[i-2]
    return cache[index]
```

```
Doc:
Sort the input array into ascending order.
Code:
def bubbleSort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            # if adjacent elements appear in descending
            # order, swap them.
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]
```

1. CodeRetriever proposes a semantic-guided method to build positive code-code pairs based on the documentation and function names.
2. CodeRetriever uses unimodal (i.e., code-code) and bimodal (i.e., text-code) contrastive learning to learn function-level code representations and achieves new SOTA results on the text-to-code search task, comparing to several strong baselines.

Xiaonan Li, Yeyun Gong, Yelong Shen, Xipeng Qiu, Hang Zhang, Bolun Yao, Weizhen Qi, Daxin Jiang, Weizhu Chen, Nan Duan. **CodeRetriever: Unimodal and Bimodal Contrastive Learning**. EMNLP 2022.

CodeRetriever (v2) with AST-Based Inverse Cloze Test



Python code with a comment

```python
# Return the sample arithmetic mean of data
def mean(data):
    n = len(data)
    return sum(data) / n
```

Query Code:

Answer Code:

Motivation: compared with applying ICT of NLP (random sampling token span of code tokens as query), AST-based ICT can generate query-answer code pairs without syntax errors.

Xiaonan Li, Daya Guo, Yeyun Gong, Yun Lin, Yelong Shen, Xipeng Qiu, Daxin Jiang, Weizhu Chen, Nan Duan. **Soft-Labeled Contrastive Pre-Training for Function-Level Code Representation**. EMNLP 2022.

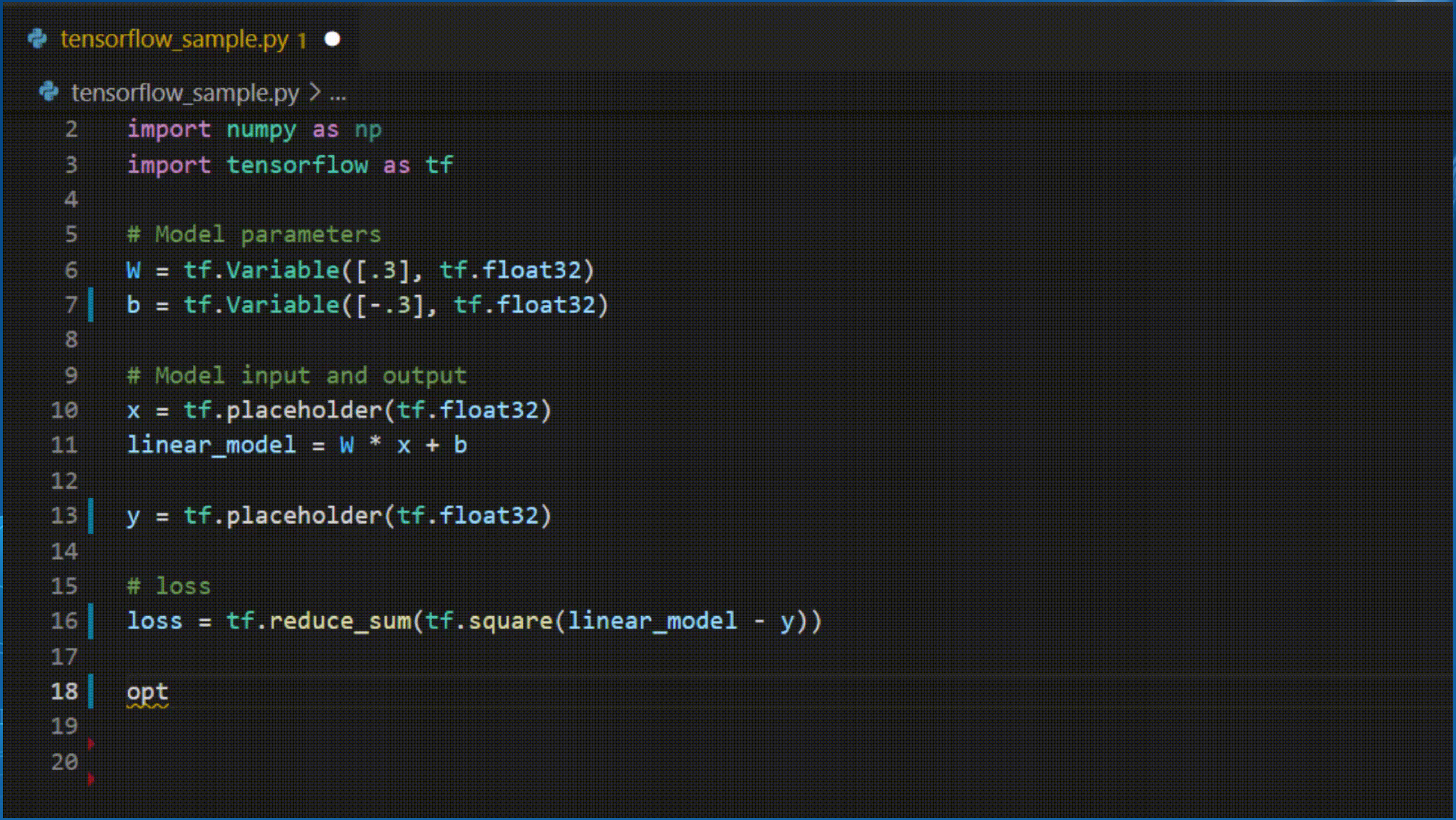
Evaluation on Code Search & Clone Detection

Code Search	**CodeSearchNet** (Husain et al., 2019)	**CoSQA** (Huang et al., 2021)	**AdvTest** (Lu et al., 2021)
CodeBERT	69.28%	27.20%	64.70%
GraphCodeBERT	73.63%	35.20%	67.50%
UniXcoder	74.35%	70.10%	41.30%
CodeRetriever (v2)	**76.56%**	**73.80%**	**44.80%**

Clone Detection	**CodeNet** (zero-shot) (Puri et al., 2021)				**POJ-104** (Mou et al., 2016)
	Ruby	**Python**	**Java**	**Overall**	**MRR**
CodeBERT	13.55%	14.39%	7.62%	11.85%	82.67%
GraphCodeBERT	17.01%	19.34%	13.31%	16.55%	85.16%
UniXcoder	29.05%	30.15%	16.12%	25.11%	90.52%
CodeRetriever (v2)	**33.72%**	**32.78%**	**18.91%**	**28.47%**	**91.90%**

Xiaonan Li, Daya Guo, Yeyun Gong, Yun Lin, Yelong Shen, Xipeng Qiu, Daxin Jiang, Weizhu Chen, Nan Duan. **Soft-Labeled Contrastive Pre-Training for Function-Level Code Representation**. EMNLP 2022.

Code Completion

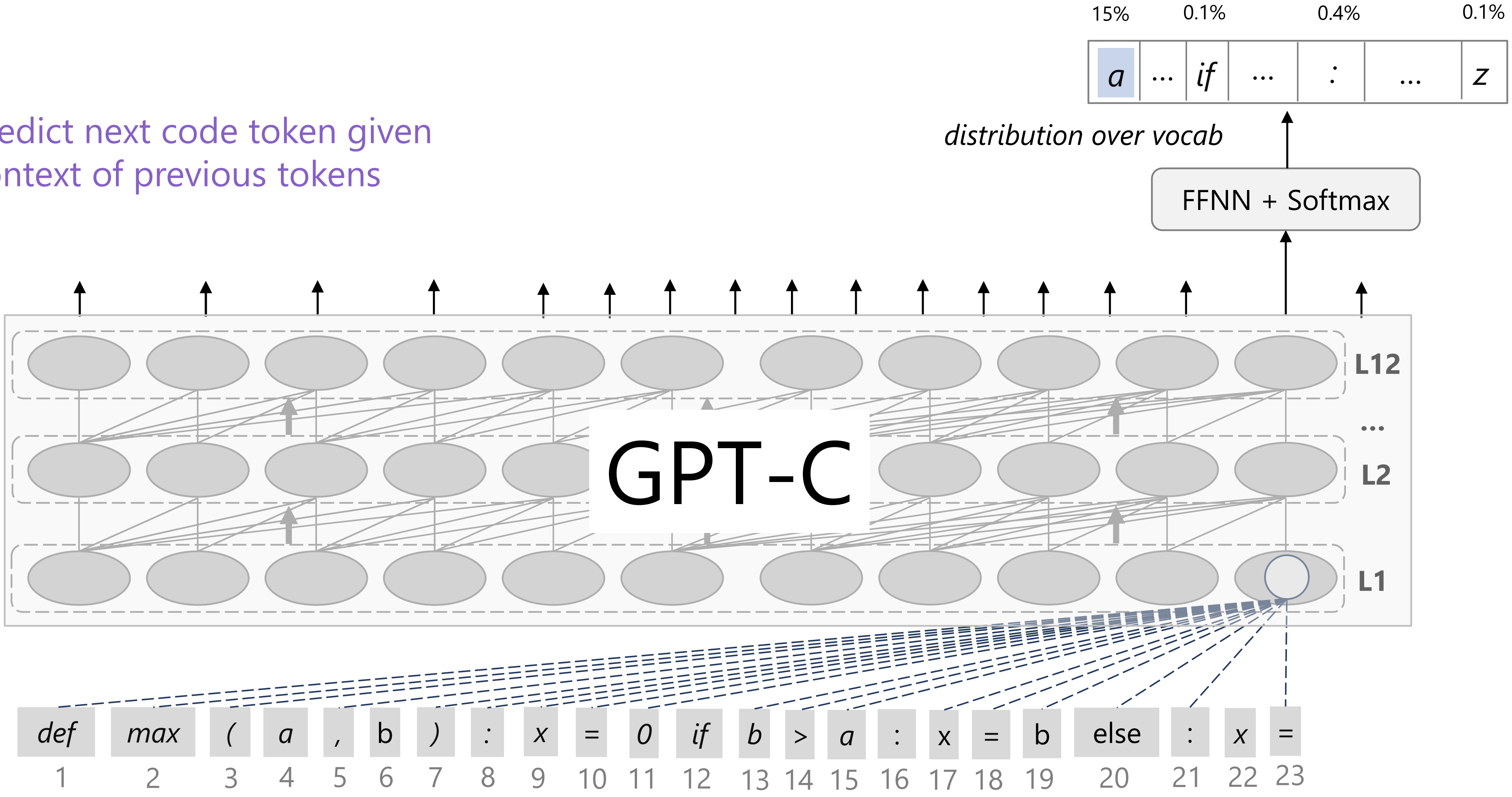


```
import numpy as np
import tensorflow as tf

# Model parameters
W = tf.Variable([.3], tf.float32)
b = tf.Variable([-.3], tf.float32)

# Model input and output
x = tf.placeholder(tf.float32)
linear_model = W * x + b

y = tf.placeholder(tf.float32)

# loss
loss = tf.reduce_sum(tf.square(linear_model - y))

opt


```

Shuai Lu, Daya Guo, Shuo Ren, Junjie Huang, Alexey Svyatkovskiy, Ambrosio Blanco, Colin Clement, Dawn Drain, Daxin Jiang, Duyu Tang, Ge Li, Lidong Zhou, Linjun Shou, Long Zhou, Michele Tufano, Ming Gong, Ming Zhou, Nan Duan, Neel Sundaresan, Shao Kun Deng, Shengyu Fu, Shujie Liu. **CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation**. NeurIPS 2021.

GPT-C (v1): Multilingual Code Completion Model

Predict next code token given context of previous tokens



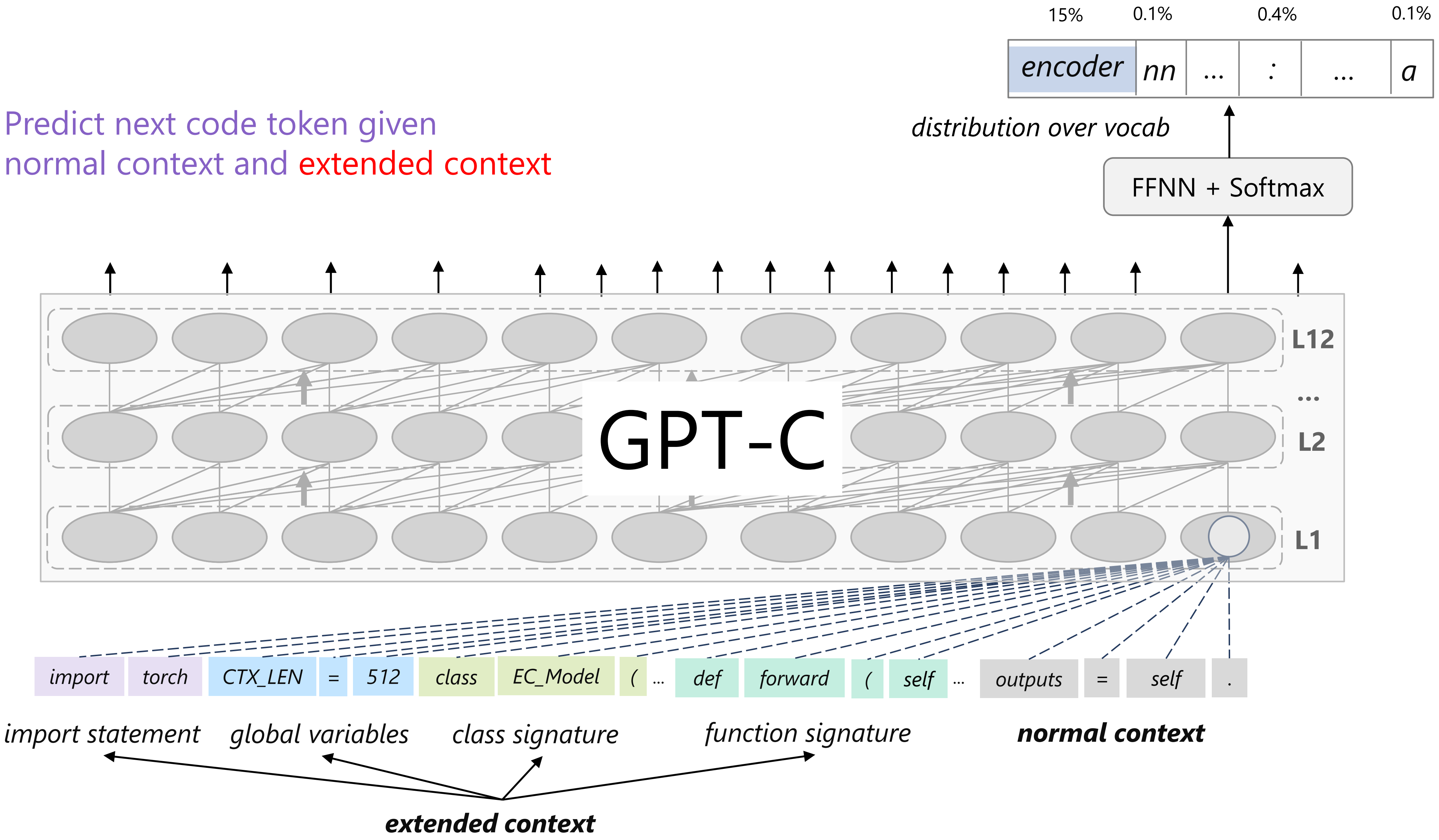
def max (a , b) : x = 0 if b > a : x = b else : x = a return x

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

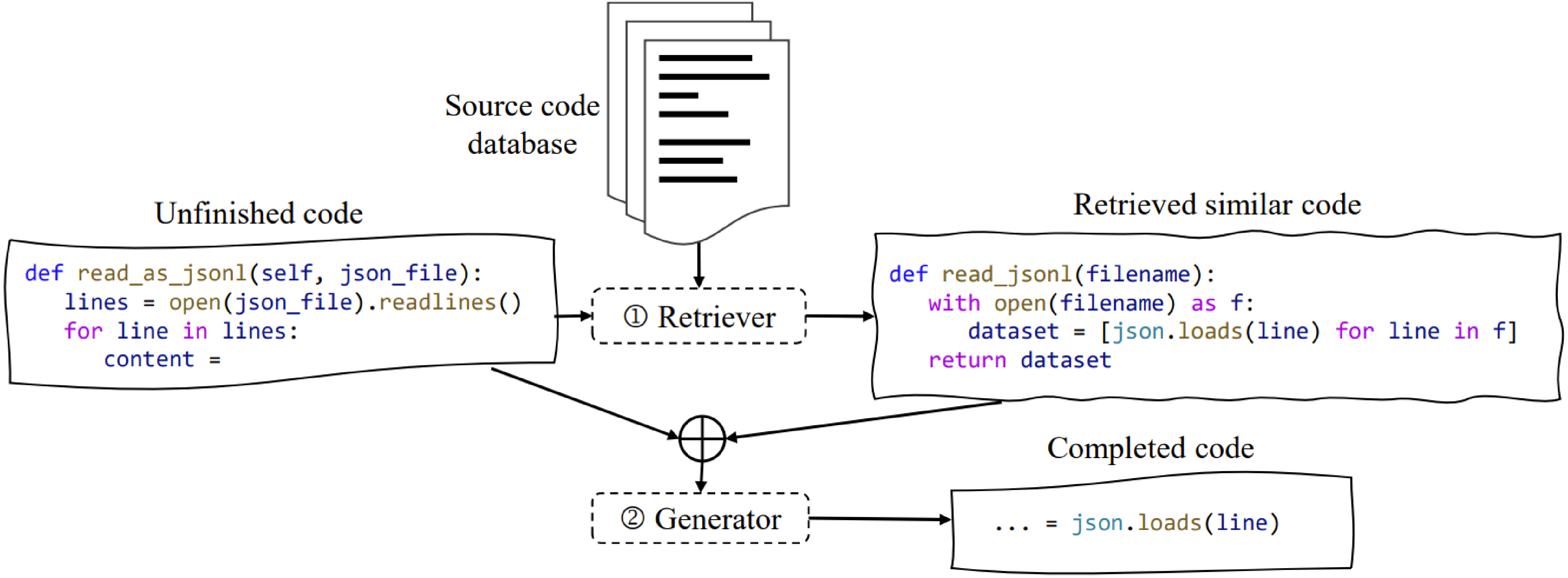
Colin Clement, Shuai Lu, Xiaoyu Liu, Michele Tufano, Dawn Drain, Nan Duan, Neel Sundaresan, Alexey Svyatkovskiy. **Long-Range Modeling of Source Code Files with eWASH: Extended Window Access by Syntax Hierarchy**. EMNLP 2021.

GPT-C (v2) with Extended Context

Predict next code token given normal context and extended context



GPT-C (v3) with Retrieved Similar Code



```
def read_as_jsonl(self, json_file):
   lines = open(json_file).readlines()
   for line in lines:
      content =
```

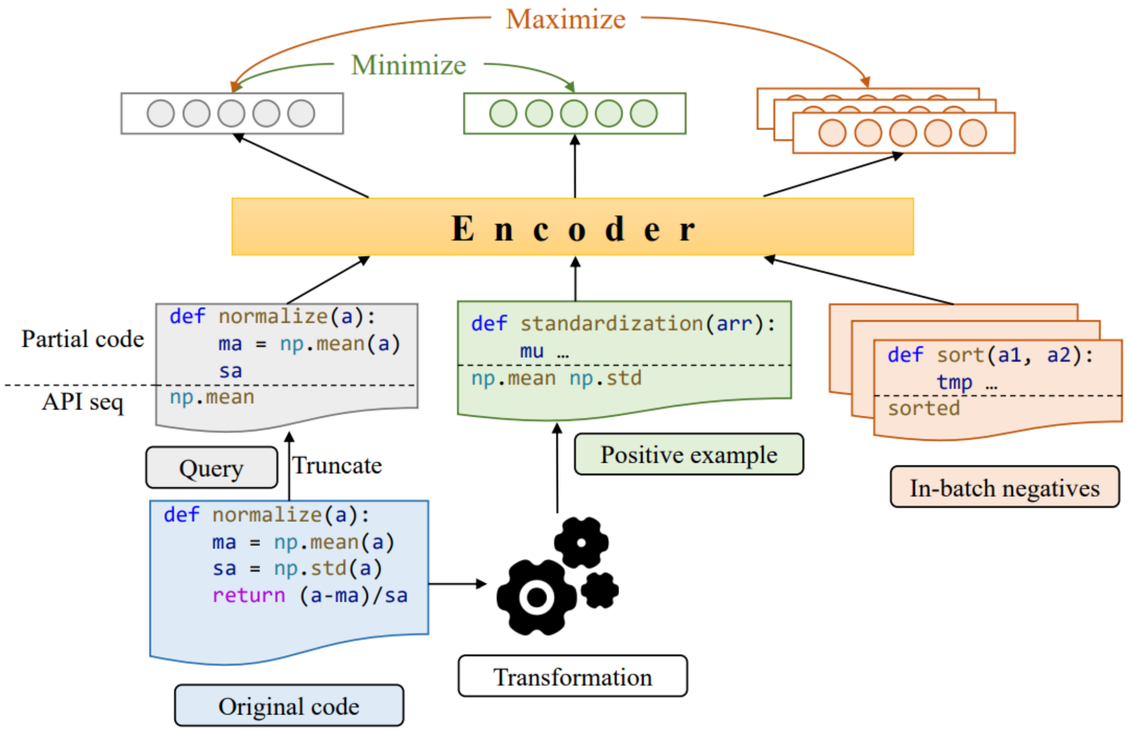
```
def read_jsonl(filename):
   with open(filename) as f:
      dataset = [json.loads(line) for line in f]
   return dataset
```

```
... = json.loads(line)
```

Shuai Lu, Nan Duan, Hojae Han, Daya Guo, Seung-won Hwang, Alexey Svyatkovskiy. **ReACC: A Retrieval-Augmented Code Completion Framework**. ACL 2022.

Code-to-Code Retrieval Pre-training

- **Goal**
  - Partial code → Similar complete code
- **Contrastive pre-training**
  - query: a random truncation of the original code + API sequence
  - + instance: the entire transformed code + API sequence
  - - instance: in-batch negatives + API sequences



```
def normalize(a):
    ma = np.mean(a)
    sa
np.mean
```

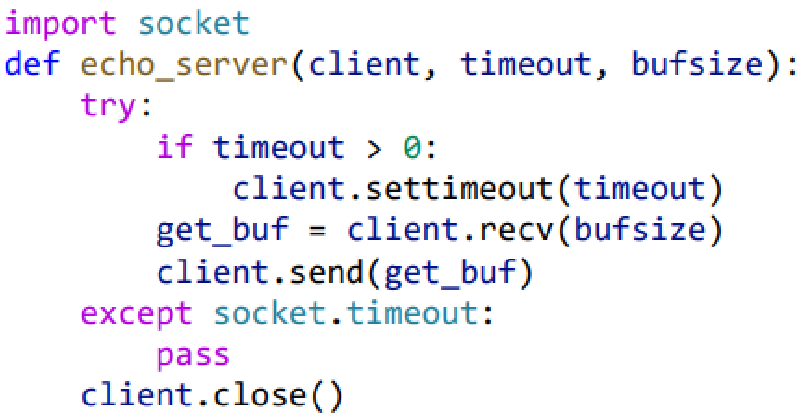
```
def standardization(arr):
    mu …
np.mean np.std
```

```
def sort(a1, a2):
    tmp …
sorted
```

```
def normalize(a):
    ma = np.mean(a)
    sa = np.std(a)
    return (a-ma)/sa
```

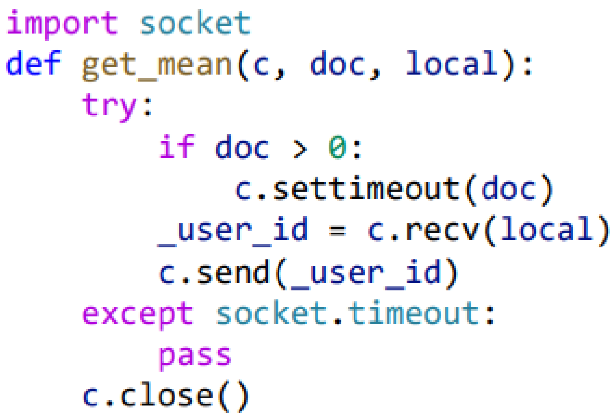
Shuai Lu, Nan Duan, Hojae Han, Daya Guo, Seung-won Hwang, Alexey Svyatkovskiy. **ReACC: A Retrieval-Augmented Code Completion Framework**. ACL 2022.

Semantic-Preserving Data Augmentation



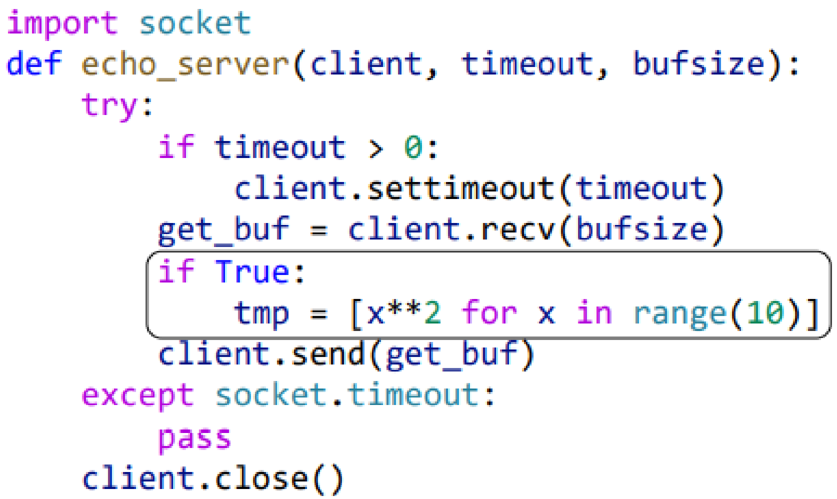
```python
import socket
def echo_server(client, timeout, bufsize):
    try:
        if timeout > 0:
            client.settimeout(timeout)
        get_buf = client.recv(bufsize)
        client.send(get_buf)
    except socket.timeout:
        pass
    client.close()
```

original python code



```python
import socket
def get_mean(c, doc, local):
    try:
        if doc > 0:
            c.settimeout(doc)
        _user_id = c.recv(local)
        c.send(_user_id)
    except socket.timeout:
        pass
    c.close()
```

After renaming all variables



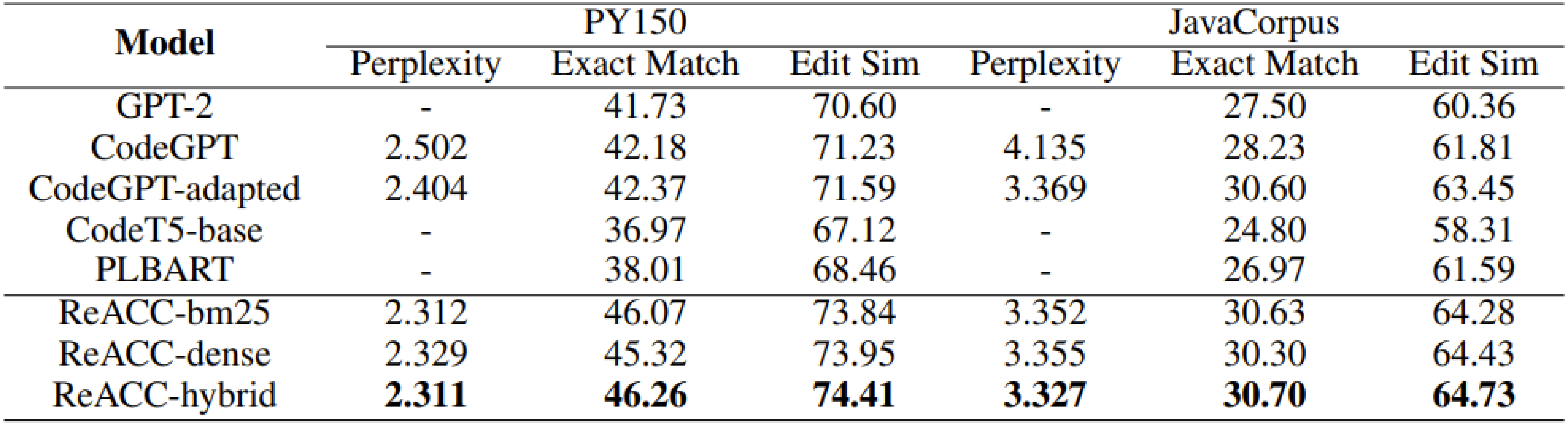
```python
import socket
def echo_server(client, timeout, bufsize):
    try:
        if timeout > 0:
            client.settimeout(timeout)
        get_buf = client.recv(bufsize)
        if True:
            tmp = [x**2 for x in range(10)]
        client.send(get_buf)
    except socket.timeout:
        pass
    client.close()
```

After inserting dead code

- **Identifier renaming** is a method of renaming an identifier with another.
- **Dead code insertion** is to insert a dead code into a code fragment at a proper location.

Shuai Lu, Nan Duan, Hojae Han, Daya Guo, Seung-won Hwang, Alexey Svyatkovskiy. **ReACC: A Retrieval-Augmented Code Completion Framework**. ACL 2022.

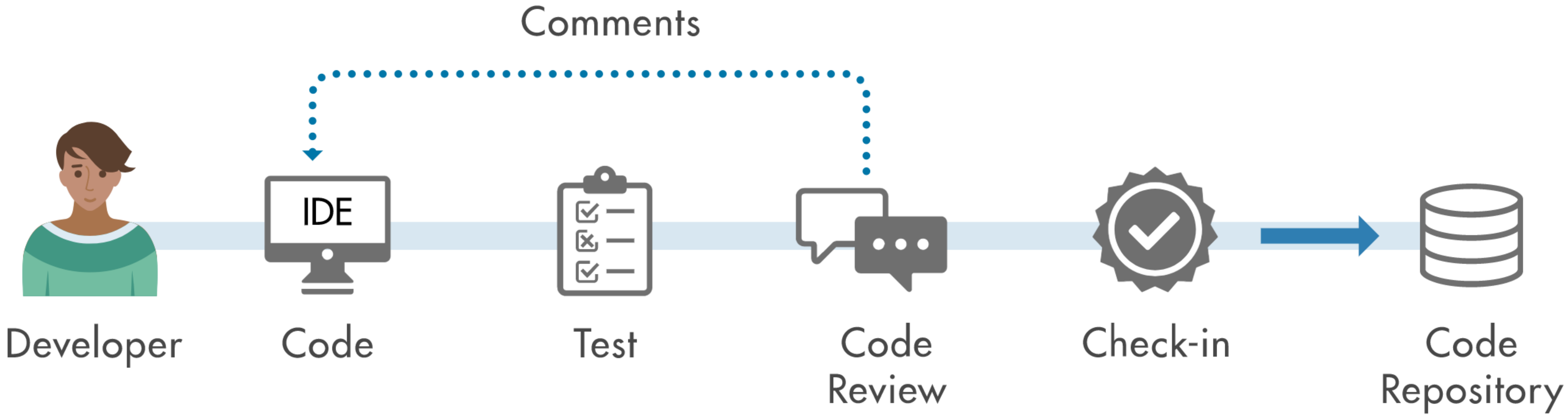
Retrieval-augmented Code Completion



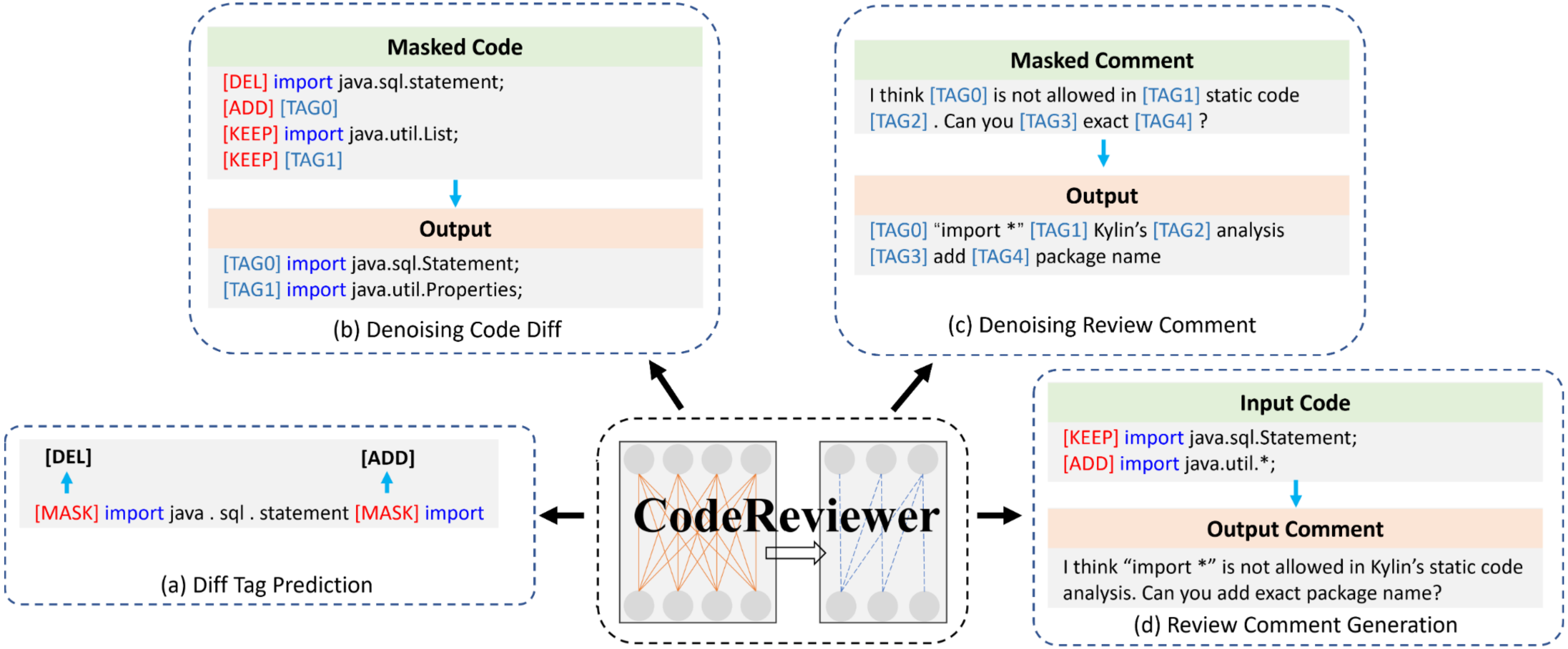
Model	PY150			JavaCorpus		
	Perplexity	Exact Match	Edit Sim	Perplexity	Exact Match	Edit Sim
GPT-2	-	41.73	70.60	-	27.50	60.36
CodeGPT	2.502	42.18	71.23	4.135	28.23	61.81
CodeGPT-adapted	2.404	42.37	71.59	3.369	30.60	63.45
CodeT5-base	-	36.97	67.12	-	24.80	58.31
PLBART	-	38.01	68.46	-	26.97	61.59
ReACC-bm25	2.312	46.07	73.84	3.352	30.63	64.28
ReACC-dense	2.329	45.32	73.95	3.355	30.30	64.43
ReACC-hybrid	**2.311**	**46.26**	**74.41**	**3.327**	**30.70**	**64.73**

Shuai Lu, Nan Duan, Hojae Han, Daya Guo, Seung-won Hwang, Alexey Svyatkovskiy. **ReACC: A Retrieval-Augmented Code Completion Framework**. ACL 2022.

Code Review & Refinement



CodeReviewer for Automating Code Review Activities



Masked Code

```
[DEL] import java.sql.statement;
[ADD] [TAG0]
[KEEP] import java.util.List;
[KEEP] [TAG1]
```

Output

```
[TAG0] import java.sql.Statement;
[TAG1] import java.util.Properties;
```

(b) Denoising Code Diff

Masked Comment

I think [TAG0] is not allowed in [TAG1] static code [TAG2] . Can you [TAG3] exact [TAG4] ?

Output

[TAG0] “import *” [TAG1] Kylin’s [TAG2] analysis [TAG3] add [TAG4] package name

(c) Denoising Review Comment

[DEL] **[ADD]**

[MASK] import java . sql . statement [MASK] import

(a) Diff Tag Prediction

Input Code

```
[KEEP] import java.sql.Statement;
[ADD] import java.util.*;
```

Output Comment

I think “import *” is not allowed in Kylin’s static code analysis. Can you add exact package name?

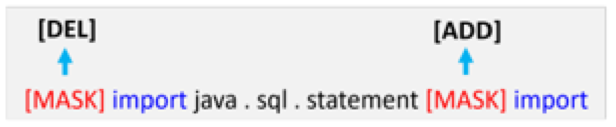
(d) Review Comment Generation

Zhiyu Li, Shuai Lu, Daya Guo, Nan Duan, Shailesh Jannu, Grant Jenks, Deep Majumder, Jared Green, Alexey Svyatkovskiy, Shengyu Fu, Neel Sundaresan. **CodeReviewer: Pre-Training for Automating Code Review Activities**. ESEC/FSE 2022.

Pre-training Tasks

- Diff Tag Prediction

$$\mathcal{L}_{DTP} = -\sum_{i} \left(y_0^{(i)} \log p_0^{(i)} + y_1^{(i)} \log p_1^{(i)} + y_2^{(i)} \log p_2^{(i)} \right)$$

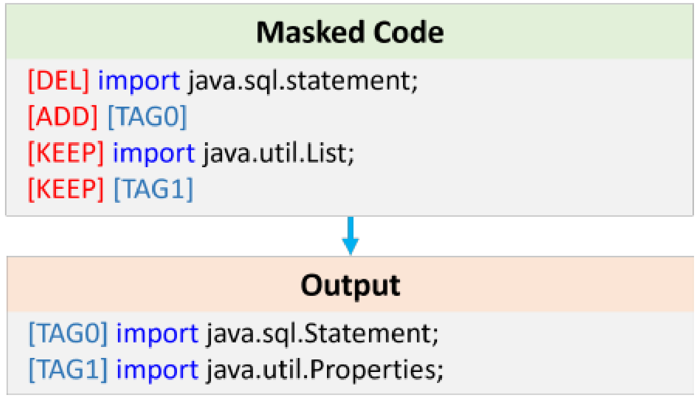


```
[MASK] import java . sql . statement [MASK] import
```

(a) Diff Tag Prediction

- Denoising Code Diff

$$\mathcal{L}_{DCD} = \sum_{t=1}^{k} -\log P_{\theta}(c_t | \mathbf{c}^{\text{mask}}, \mathbf{c}_{<t})$$



Masked Code

```
[DEL] import java.sql.statement;
[ADD] [TAG0]
[KEEP] import java.util.List;
[KEEP] [TAG1]
```

Output

```
[TAG0] import java.sql.Statement;
[TAG1] import java.util.Properties;
```

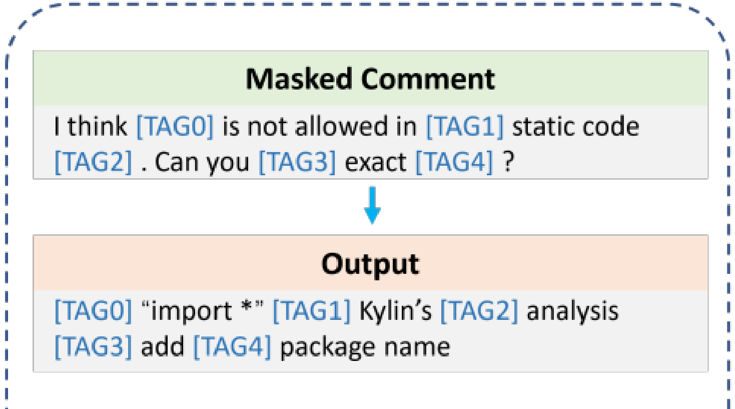
(b) Denoising Code Diff

Zhiyu Li, Shuai Lu, Daya Guo, Nan Duan, Shailesh Jannu, Grant Jenks, Deep Majumder, Jared Green, Alexey Svyatkovskiy, Shengyu Fu, Neel Sundaresan. **CodeReviewer: Pre-Training for Automating Code Review Activities**. ESEC/FSE 2022.

Pre-training Tasks

- Denoising Review Comment

$$\mathcal{L}_{DRC} = \sum_{t=1}^{k} -\log P_{\theta}(w_t | \mathbf{w}^{\text{mask}}, \mathbf{w}_{<t})$$



Masked Comment

I think [TAG0] is not allowed in [TAG1] static code [TAG2] . Can you [TAG3] exact [TAG4] ?

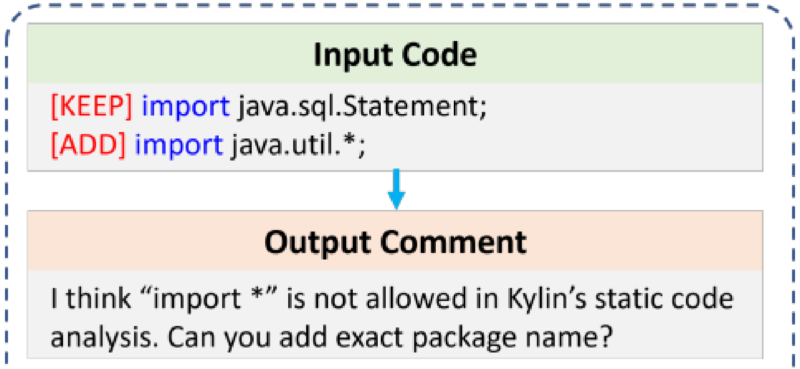
Output

[TAG0] “import *” [TAG1] Kylin’s [TAG2] analysis [TAG3] add [TAG4] package name

(c) Denoising Review Comment

- Review Comment Generation

$$\mathcal{L}_{RCG} = \sum_{t=1}^{k} -\log P(w_t | \mathbf{c}, \mathbf{w}_{<t})$$



Input Code

[KEEP] import java.sql.Statement;
[ADD] import java.util.*;

Output Comment

I think “import *” is not allowed in Kylin’s static code analysis. Can you add exact package name?

(d) Review Comment Generation

Zhiyu Li, Shuai Lu, Daya Guo, Nan Duan, Shailesh Jannu, Grant Jenks, Deep Majumder, Jared Green, Alexey Svyatkovskiy, Shengyu Fu, Neel Sundaresan. **CodeReviewer: Pre-Training for Automating Code Review Activities.** ESEC/FSE 2022.

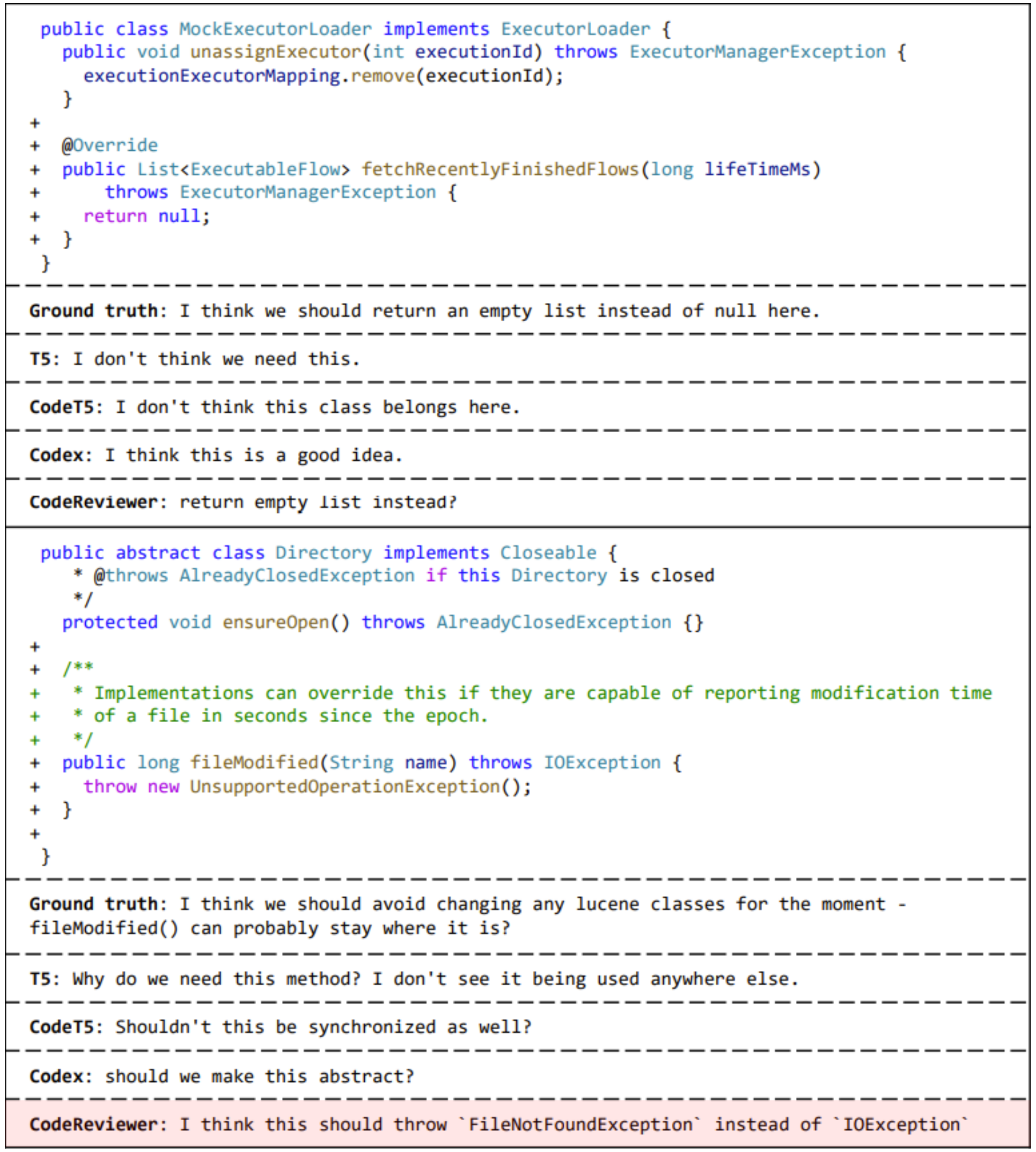
Code Review Dataset

1. CodeReviewer is pre-trained on the pull requests crawled from GitHub in 9 programming languages and establishes a benchmark dataset for code review activities.
2. Collected from projects who containing more than 1000 PRs.

Language	Ruby	Go	Php	Js	Python	Java	C++	C#	C	**Total**
# Project	72	146	92	194	195	175	133	77	77	**1.2M**
# (Diff, NL)	126k	410k	157k	293k	518k	467k	202k	199k	110k	**2.5M**
# Diff	90k	728k	215k	425k	887k	876k	474k	324k	292k	**4.3M**

Zhiyu Li, Shuai Lu, Daya Guo, Nan Duan, Shailesh Jannu, Grant Jenks, Deep Majumder, Jared Green, Alexey Svyatkovskiy, Shengyu Fu, Neel Sundaresan. **CodeReviewer: Pre-Training for Automating Code Review Activities.** ESEC/FSE 2022.

Examples



```
 public class MockExecutorLoader implements ExecutorLoader {
   public void unassignExecutor(int executionId) throws ExecutorManagerException {
     executionExecutorMapping.remove(executionId);
   }
+
+  @Override
+  public List<ExecutableFlow> fetchRecentlyFinishedFlows(long lifeTimeMs)
+      throws ExecutorManagerException {
+    return null;
+  }
 }
```

Ground truth: I think we should return an empty list instead of null here.

T5: I don't think we need this.

CodeT5: I don't think this class belongs here.

Codex: I think this is a good idea.

CodeReviewer: return empty list instead?

```
 public abstract class Directory implements Closeable {
    * @throws AlreadyClosedException if this Directory is closed
    */
   protected void ensureOpen() throws AlreadyClosedException {}
+
+  /**
+   * Implementations can override this if they are capable of reporting modification time
+   * of a file in seconds since the epoch.
+   */
+  public long fileModified(String name) throws IOException {
+    throw new UnsupportedOperationException();
+  }
+
 }
```

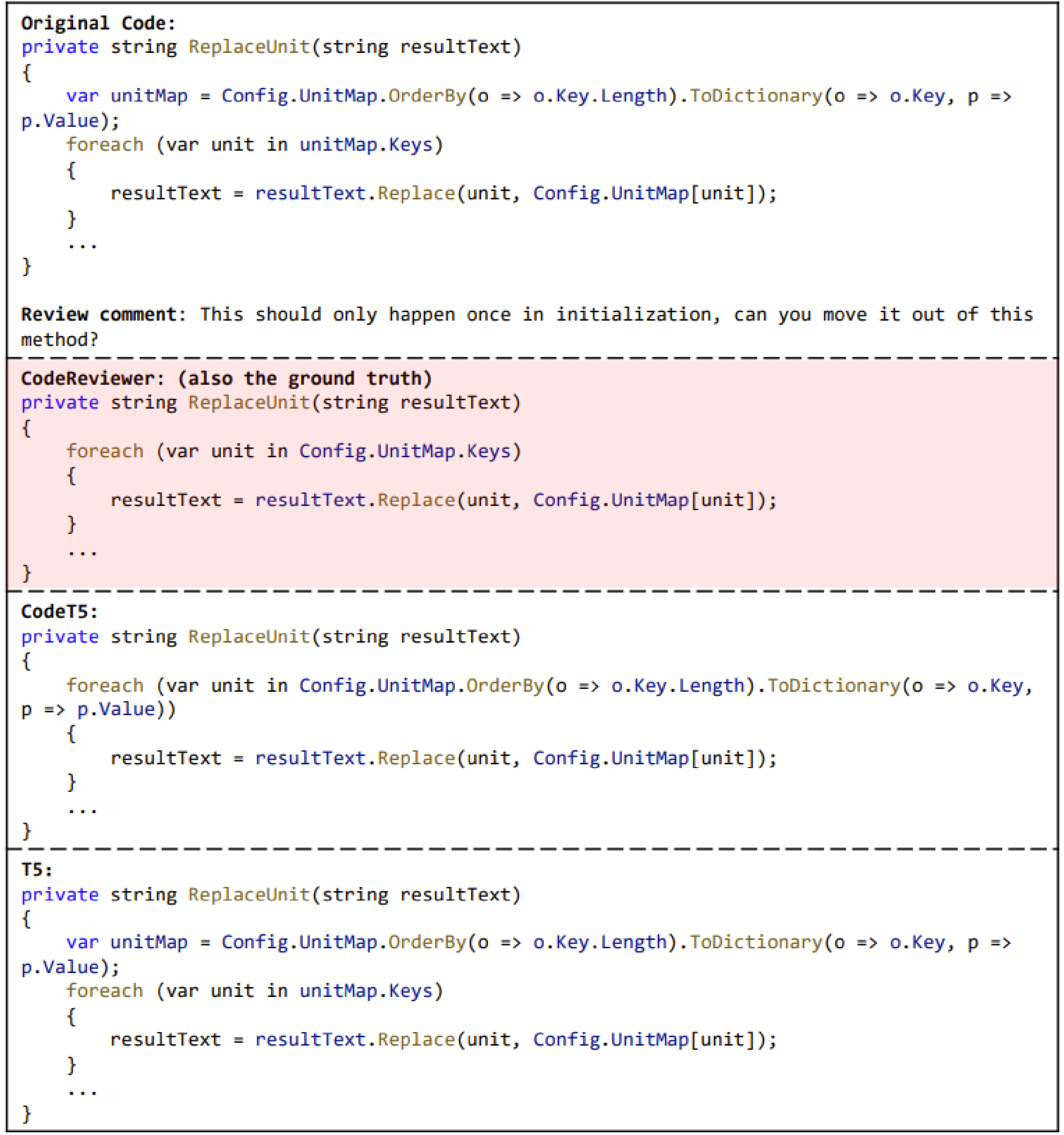
Ground truth: I think we should avoid changing any lucene classes for the moment - fileModified() can probably stay where it is?

T5: Why do we need this method? I don't see it being used anywhere else.

CodeT5: Shouldn't this be synchronized as well?

Codex: should we make this abstract?

CodeReviewer: I think this should throw `FileNotFoundException` instead of `IOException`



Original Code:

```
private string ReplaceUnit(string resultText)
{
    var unitMap = Config.UnitMap.OrderBy(o => o.Key.Length).ToDictionary(o => o.Key, p =>
p.Value);
    foreach (var unit in unitMap.Keys)
    {
        resultText = resultText.Replace(unit, Config.UnitMap[unit]);
    }
    ...
}
```

Review comment: This should only happen once in initialization, can you move it out of this method?

CodeReviewer: (also the ground truth)

```
private string ReplaceUnit(string resultText)
{
    foreach (var unit in Config.UnitMap.Keys)
    {
        resultText = resultText.Replace(unit, Config.UnitMap[unit]);
    }
    ...
}
```

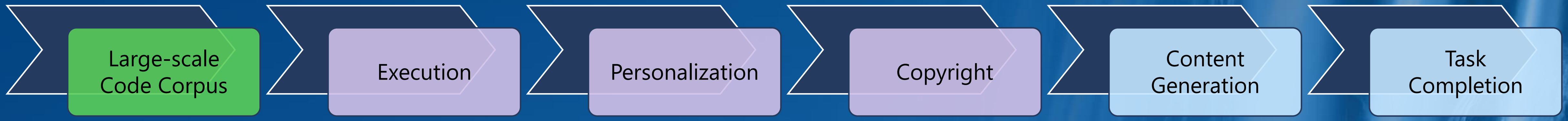
CodeT5:

```
private string ReplaceUnit(string resultText)
{
    foreach (var unit in Config.UnitMap.OrderBy(o => o.Key.Length).ToDictionary(o => o.Key,
p => p.Value))
    {
        resultText = resultText.Replace(unit, Config.UnitMap[unit]);
    }
    ...
}
```

T5:

```
private string ReplaceUnit(string resultText)
{
    var unitMap = Config.UnitMap.OrderBy(o => o.Key.Length).ToDictionary(o => o.Key, p =>
p.Value);
    foreach (var unit in unitMap.Keys)
    {
        resultText = resultText.Replace(unit, Config.UnitMap[unit]);
    }
    ...
}
```

Summary



Large-scale Code Corpus

Execution

Personalization

Copyright

Content Generation

Task Completion

- Visual Studio
- VSCode
- GitHub
- Bing
- Vertical Tasks

- Execution-based Code Pre-training
- Interactive Code Models
- Personalized Models for Code
  - Personalized variables, functions, APIs, coding styles, etc.
- Responsible Models for Code
  - Traceable predictions

- Code-centric Content Generation
- Code-centric Task Completion

谢谢！