Diet Code Is Healthy: Simplifying Programs for Pre-trained Models of Code

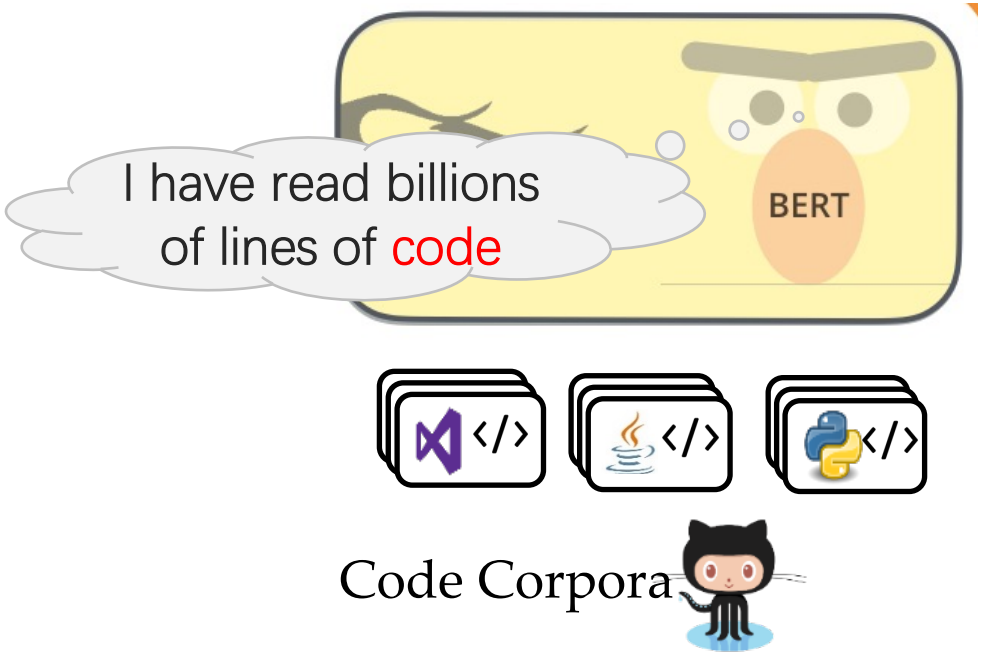
- Zhaowei Zhang[1], Hongyu Zhang[2], Beijun Shen[1], Xiaodong Gu[1]

[1]School of Software, Shanghai Jiao Tong University
[2]The University of Newcastle

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

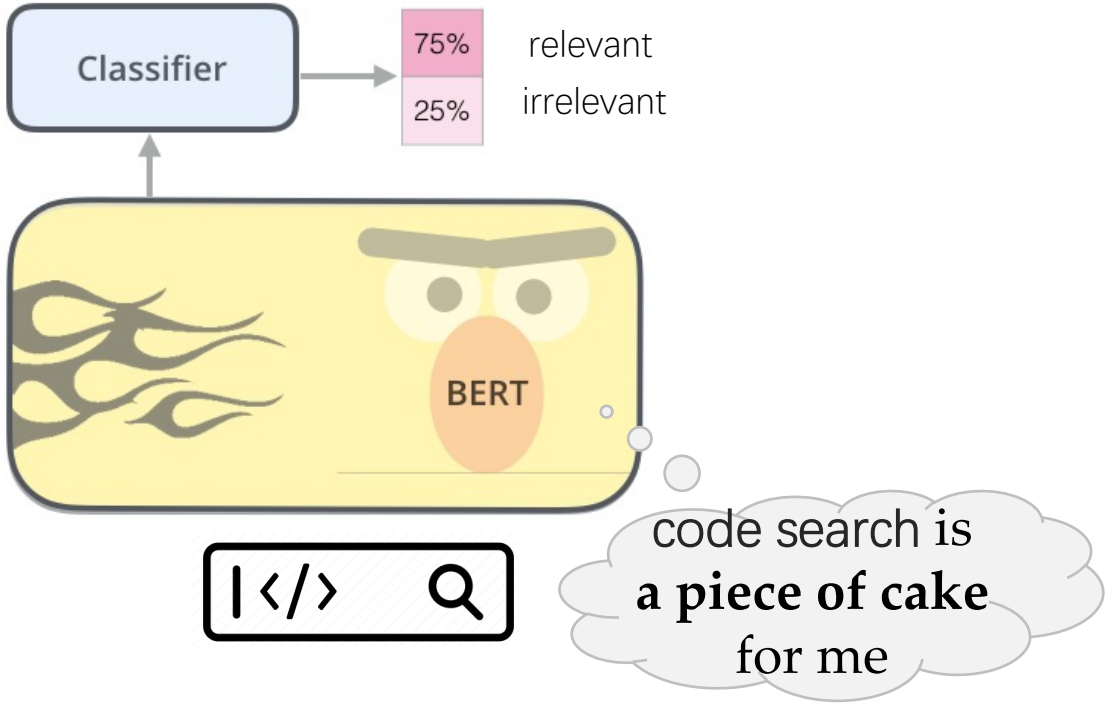
Pre-trained Models of Code

Unsupervised pre-training on large-scale code corpora



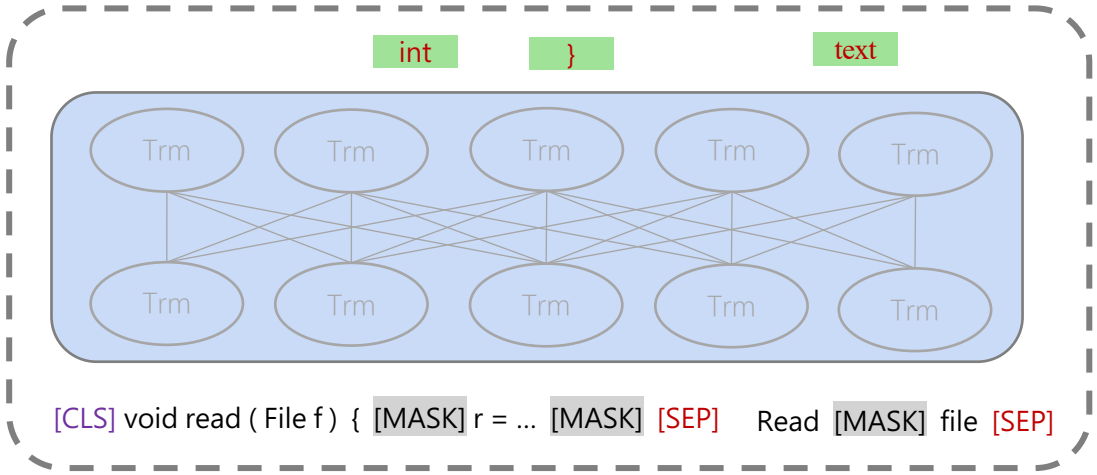
Phase1: Pre-Training

Fine-tuning on small-scale task-specific data



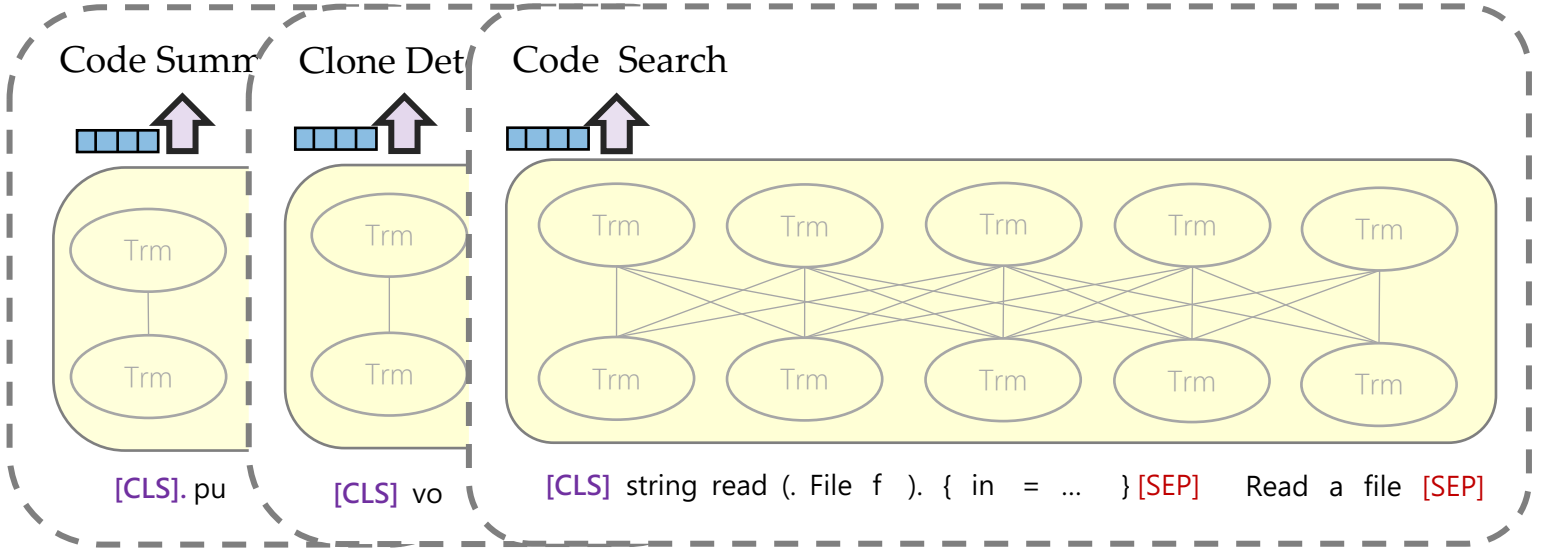
Phase2 : Fine-Tuning

CodeBERT



Pre-training
(MLM+RTD)

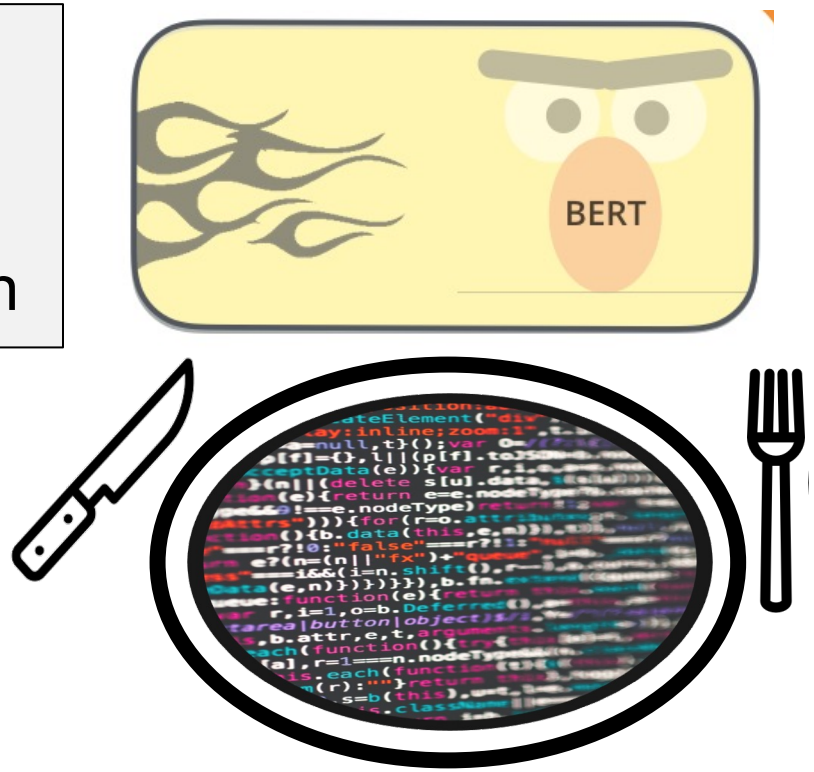
Fine-tuning



Problems with Pre-Trained Language Models

Main Challenge:

- Long Source Code
- Heavy in Computation



CodeBERT

- ~300 tokens per method
- 125 million parameters
- 12 hours pretraining

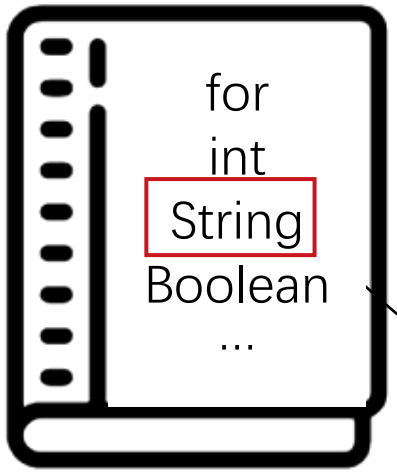
Empirical Study

RQ1: What critical tokens does CodeBERT learn about code?

RQ2: What critical statements does CodeBERT learn about code?

Critical Tokens Learned by CodeBERT

- **Methodology**



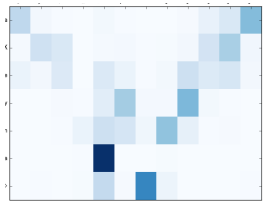
Code Corpora

... public String readFile (String name) ...

... public void setName (String name) ...

⋮

... int sort (int [] array) { ... String …}



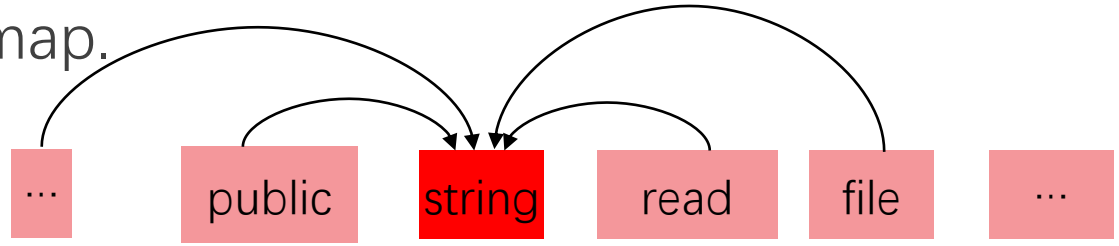
Attention

... public String readFile (...

Critical Tokens Learned by CodeBERT

- **Methodology**

For each token, summarize the attention weights assigned by other tokens in the attention map.

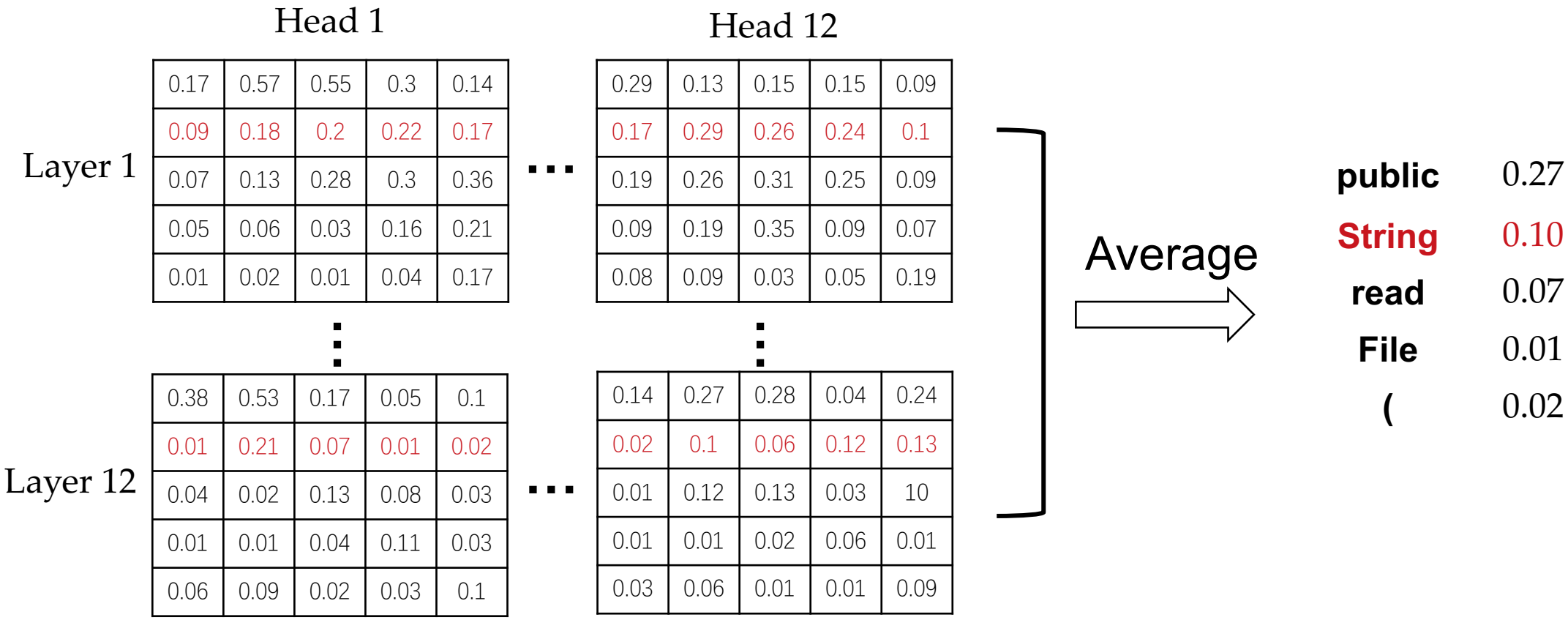


	...	public	String	read	File	(	...
...	0.17	0.57	0.55	0.3	0.14		
public	0.09	0.18	0.2	0.22	0.17		
String	0.07	0.13	0.28	0.3	0.36		
read	0.05	0.06	0.03	0.16	0.21		
File	0.01	0.02	0.01	0.04	0.17		
(							
...							

Critical Tokens Learned by CodeBERT

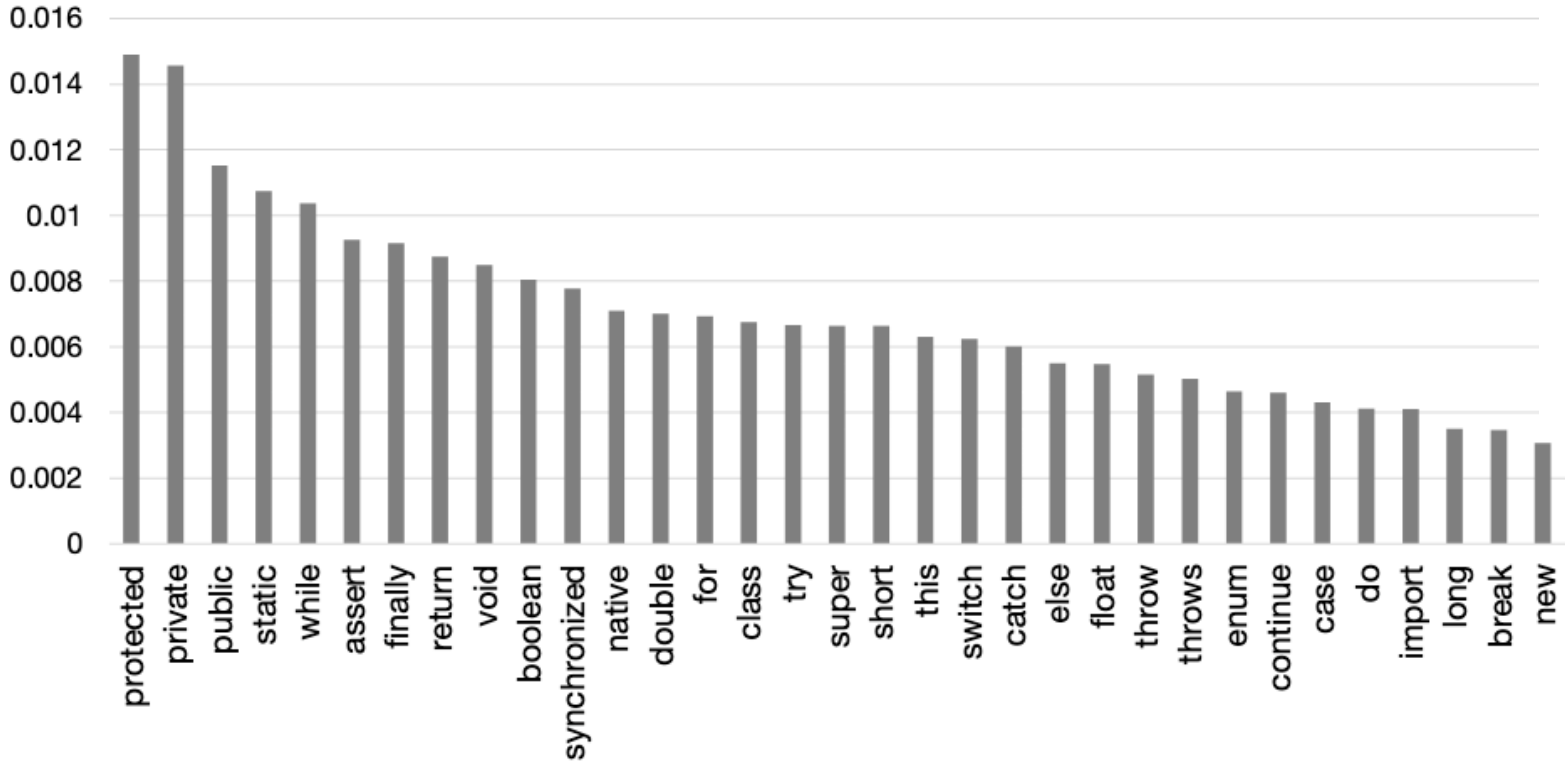
- **Methodology**

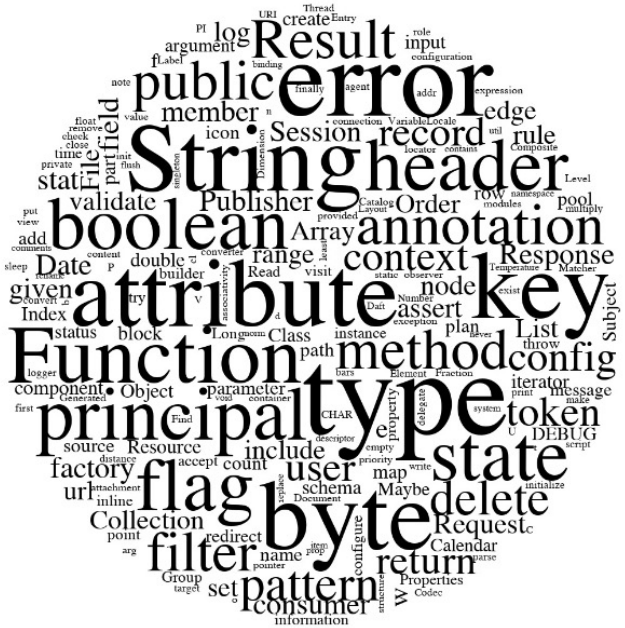
  Take the average of all layers and heads



Critical Tokens Learned by CodeBERT

- **Results**

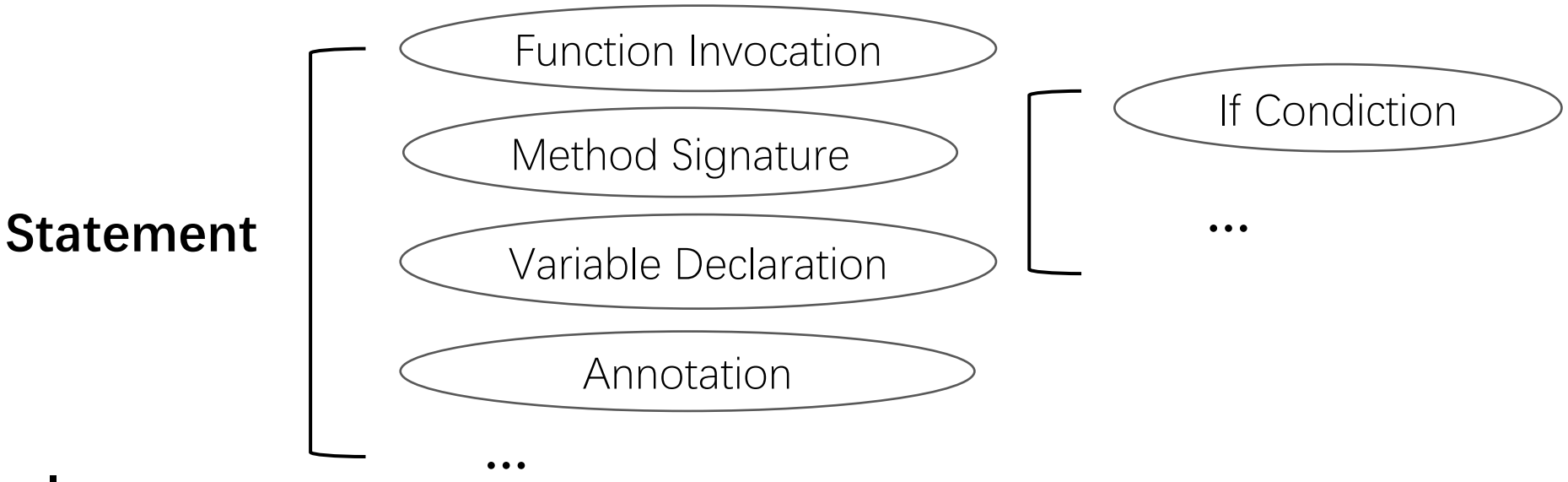




Critical Statements Learned by CodeBERT

- **Methodology**

1. Statement Categorization



Example:

```
public String readFile (String filename)
```

→ **Method Signature**

Critical Statements Learned by CodeBERT

- **Methodology**

2. Calculate Statement Attention

$$a(S) = \sum_{t \in S} w(t) \cdot a(t)$$

attention received for token t in S

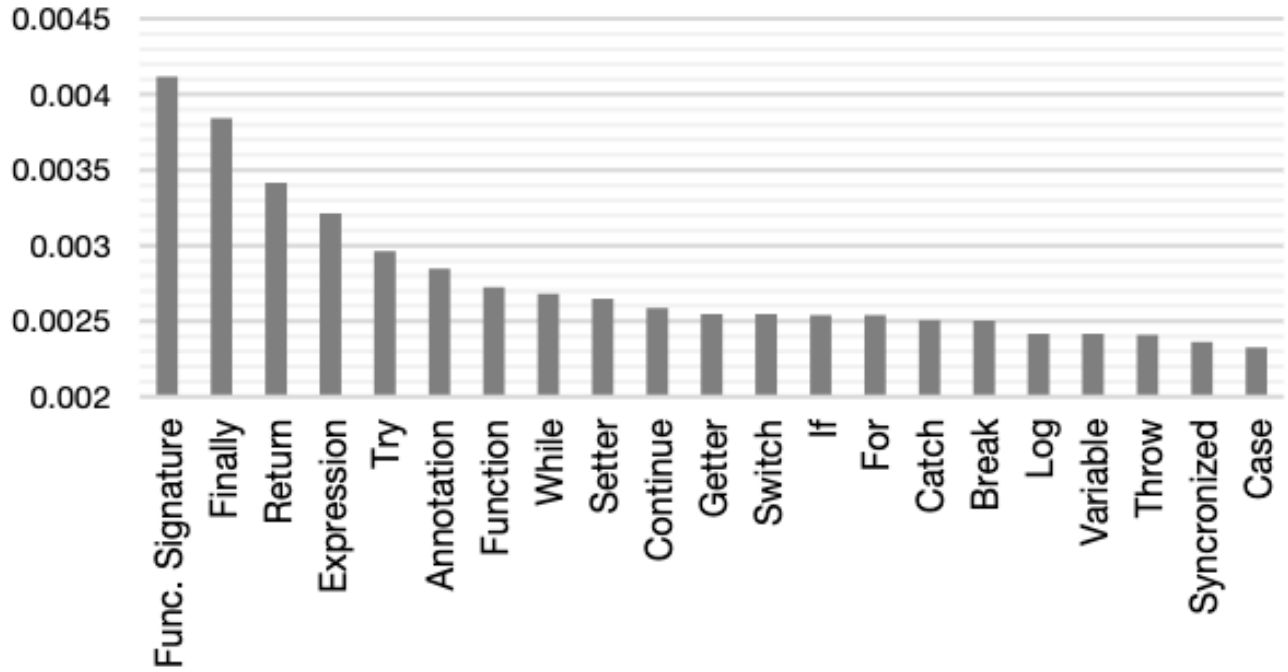
the global importance of token t in the code corpus

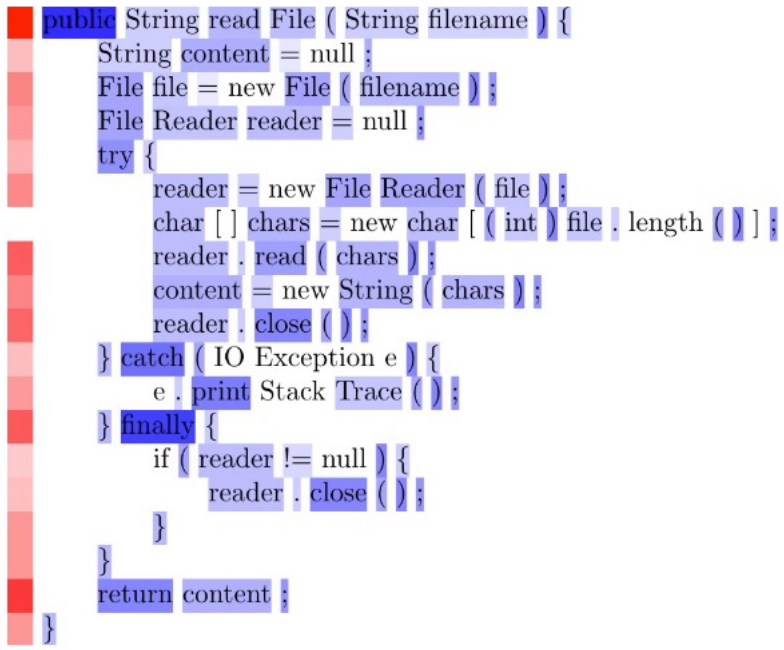
$$w(t) = Softmax_{t \in S}(a(t))$$

Critical Statements Learned by CodeBERT

- **Results**

Attention weights of various types of Java statements learned by CodeBERT





A heatmap of attention weights for Java statements and tokens.

DietCode – Simplifying Code for PLMs

Simplify source code based on their attention weights.

Input Source Code

```java
public String readFile (String filename) {
    String content = null;
    File file = new File(filename);
    FileReader reader = null;
    try {
        reader = new FileReader(file);
        char[] chars = new char[(int)file.length()];
        reader.read(chars);
        content = new String(chars);
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if(reader != null){
            reader.close();
        }
    }
    return content;
}
```

Code Pruning

Simplified Code

```java
public String readFile (String filename){
    File file = new File(filename);
    FileReader
    try {
        reader = new FileReader(file);
        reader.read(chars);
        content = new String(chars);
        reader.close();
    } finally {}
    return content;
}
```

PLM

Statement Pruning

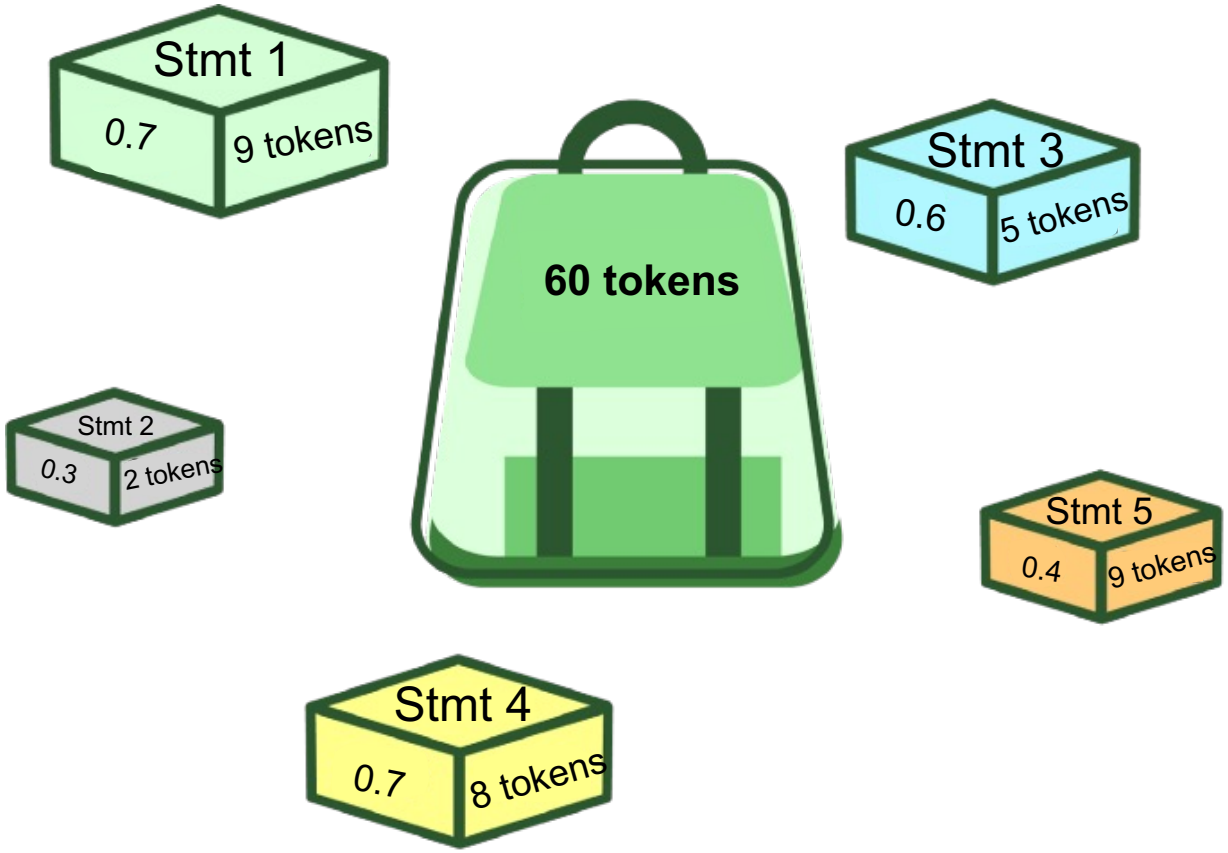
1. Prune statements based on their length and attention with a dynamical programming algorithm.
2. Repeat step 1 until the sequence length approaches the target 60.

```java
public String readFile (String filename) {
    String content = null;
    File file = new File(filename);
    FileReader reader = null;
    try {
        reader = new FileReader(file);
        char[] chars = new char[(int)file.length()];
        reader.read(chars);
        content = new String(chars);
        reader.close();
    } catch (IOException e) {
        e.printStackTrace();
    } finally {
        if(reader != null){
            reader.close();
        }
    }
    return content;
}
```

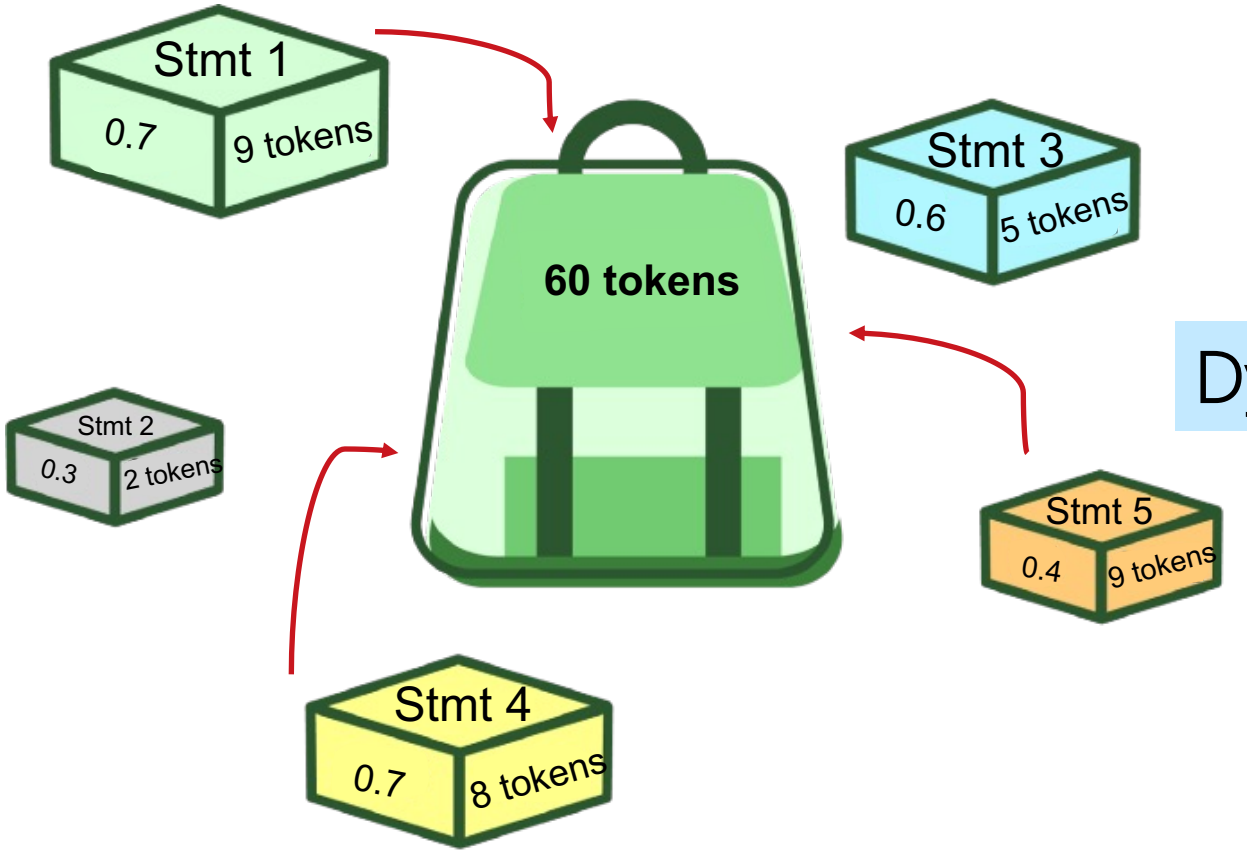
Statement Pruning

```java
public String readFile(String filename){
    String content = null;
    File file = new File(filename);
    FileReader reader = null;
    try {
        reader = new FileReader(file);
        reader.read(chars);
        content = new String(chars);
        reader.close();
    } catch (IOException e) {
    } finally {}
    return content;
}
```

Combining Both – a 0-1 Knapsack Problem



Combining Both – a 0-1 Knapsack Problem



Dynamic Programming

Token Pruning

1. Remove the lowest attentioned tokens from the lowest attentioned statements (e.g., variable initialization).
2. Repeat step 1 until the sequence length reach the target of 60.

```
1  public String readFile(String filename){
2      String content = null;
3      File file = new File(filename);
4      FileReader reader = null;
5      try {
6          reader = new FileReader(file);
7          reader.read(chars);
8          content = new String(chars);
9          reader.close();
10     } catch (IOException e) {
11     } finally {}
12     return content;
13 }
```

Token Pruning

```
1  public String readFile (String filename){
2      File file = new File(filename);
3      FileReader
4      try {
5          reader = new FileReader(file);
6          reader.read(chars);
7          content = new String(chars);
8          reader.close();
9      } finally {}
10     return content;
11 }
```

Evaluation

RQ3: How effective is DietCode in program simplification?

RQ4: How effective is DietCode under different relative length?

RQ5: What is the effect of different pruning strategies?

Datasets

- Dataset for fine-tuning and testing

Corpus	Fine-tuning	Testing
CodeSearchNet (Java)	908,886	1,000,000
CodeSearchNet (Python)	824,343	1,000,000

Metrics

- Code Search
  - $\text{MRR}=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{\text{Rank(i)}}$
- Code Summarization
  - $\text{BLEU\_4} = \frac{1}{4}\sum_{n=1}^{4}\frac{\sum_{n-gram\in c} Count(n-gram\in r)}{\sum_{n-gram\in c} Count(n-gram)}$
- Relative length:
  - $RL = \frac{|C_p|}{|C|} \times 100\%$
- FLOPs: floating point operations

Experimental Results

- Performance on Code Search

Language	Model	Relative Length	FT Time	FLOPs	MRR
Java	SelfAttn	100%	22.83h	16.99G	0.59
	Roberta	100%	23.85h	16.99G	0.67
	CodeBERT	100%	20.82h	16.99G	0.74
	DietCode	60%	11.08h	10.19G	0.71
Python	SelfAttn	100%	17.78h	16.99G	0.69
	Roberta	100%	19.32h	16.99G	0.81
	CodeBERT	100%	17.92h	16.99G	0.84
	DietCode	60%	9.62h	10.19G	0.81

DietCode achieves comparable accuracy as CodeBERT with only 60% computation cost.

Experimental Results

- Performance on Code Summarization

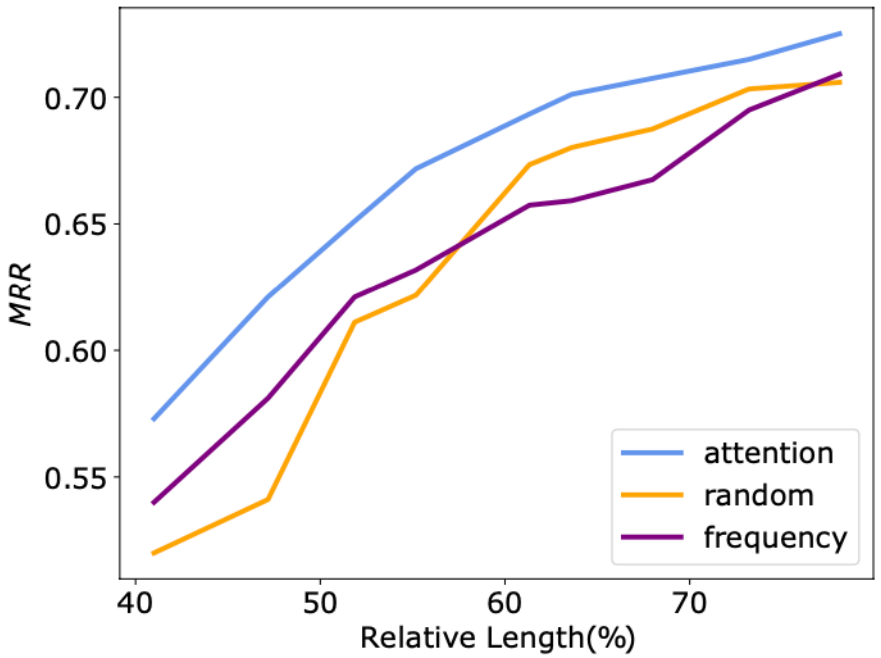
Language	Model	Relative Length	FT Time	FLOPs	MRR
Java	SelfAttn	100%	13.28h	24.33G	12.26
	Roberta	100%	15.42h	24.33G	16.47
	CodeBERT	100%	13.82h	24.33G	17.65
	DietCode	**60%**	**8.18h**	**15.33G**	**17.29**
Python	SelfAttn	100%	9.70h	24.33G	15.81
	Roberta	100%	10.63h	24.33G	18.14
	CodeBERT	100%	8.32hh	24.33G	19.04
	DietCode	**60%**	**5.35hh**	**15.33G**	**17.08**

DietCode achieves comparable accuracy as CodeBERT with 60% computation cost.

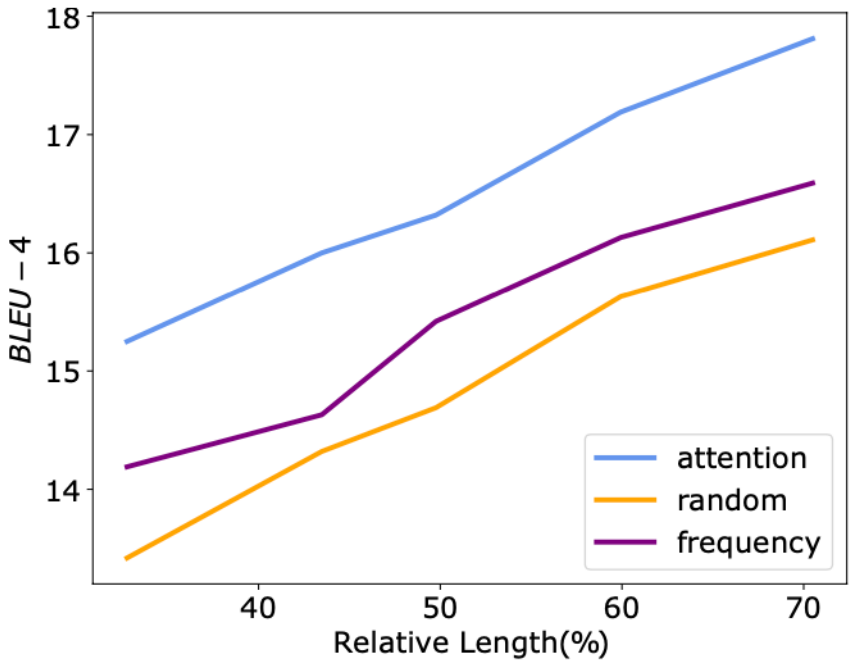
Experimental Results

- Other Pruning Strategies

Frequency: removes uncommon tokens
Random: randomly dropping tokens



(a) results on code search



(b) results on code summarization

Experimental Results

- Performance of CodeT5 with different pruning strategies on code search

Language	Prune Strategy	Relative Length	FT Time	FLOPs	MRR
Java	None	100%	16.83h	16.99G	0.72
	Attention	60%	9.30h	10.19G	0.71
	Dropout	60%	9.25h	10.19G	0.68
	Frequency	60%	8.97h	10.19G	0.66
Python	None	100%	17.61h	16.99G	0.84
	Attention	60%	8.31h	10.19G	0.81
	Dropout	60%	9.33h	10.19G	0.80
	Frequency	60%	8.67h	10.19G	0.79

DietCode achieves a comparable performance against vanilla CodeT5 and outperforms other strategies significantly.

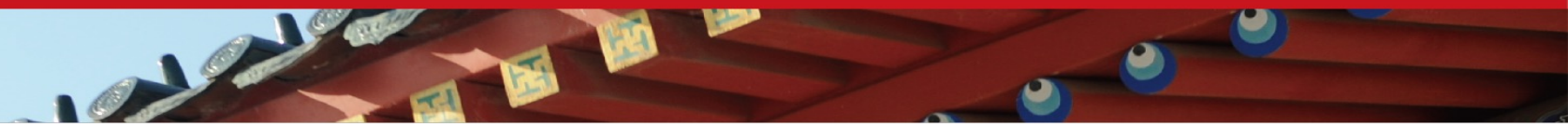
Conclusion

DietCode – lightweight leverage of pre-trained models.

- Empirical analysis of the critical information learned by CodeBERT
- A program simplification approach for PLMs with the advantage of computational efficiency.

Future Work

- incorporate code structures based on our empirical study.
- other software engineering tasks.



Thank You!

Q&A