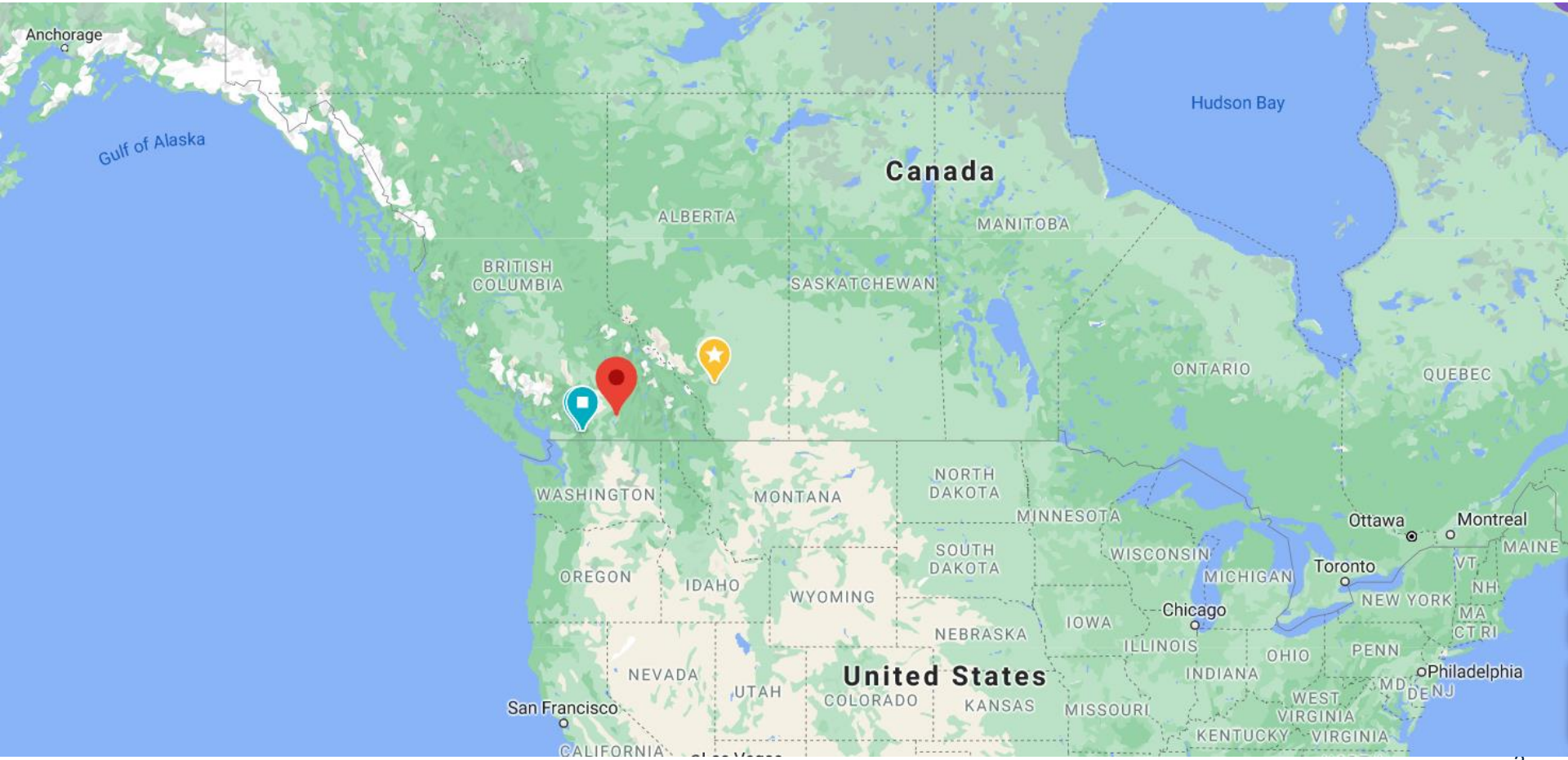
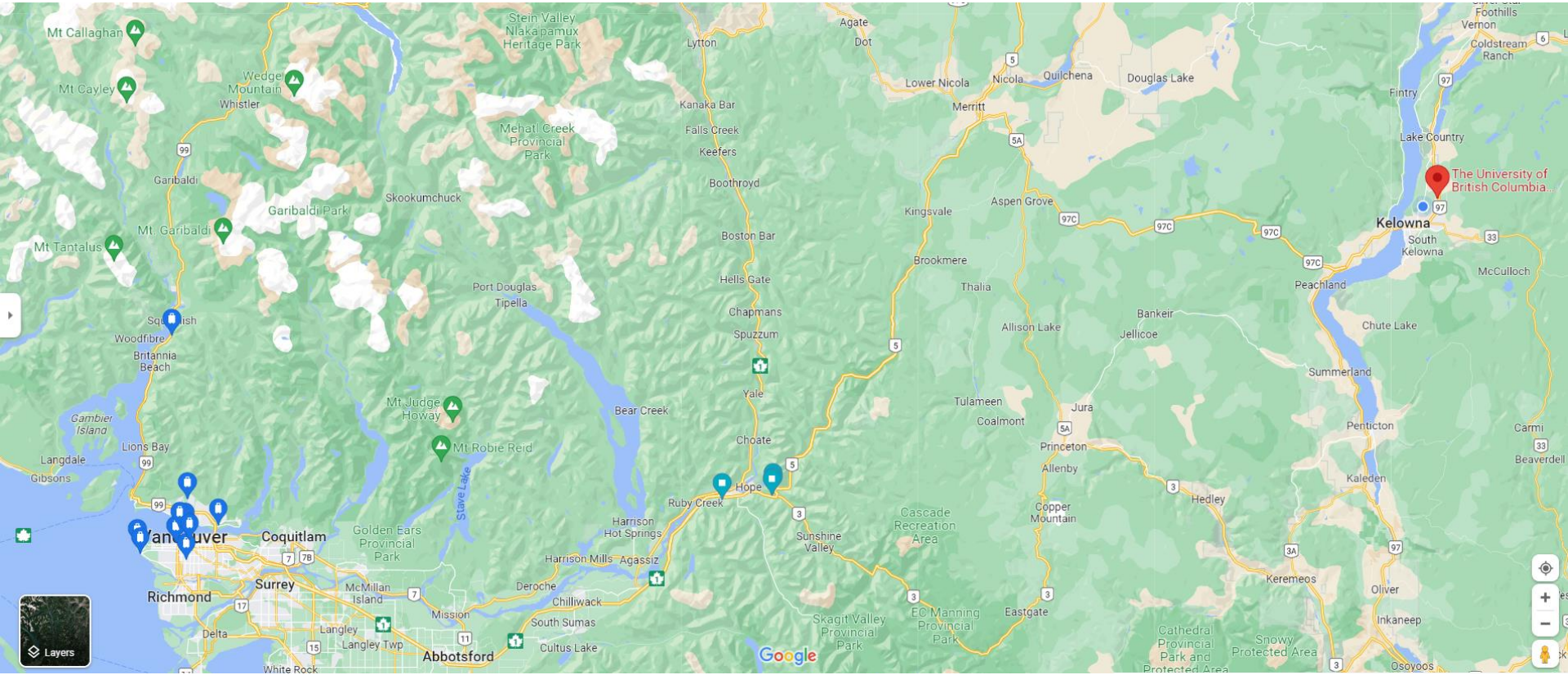
EXPLOITING THE LEARNED KNOWLEDGE OF LANGUAGE MODELS USING ADAPTERS

FATEMEH H. FARD
ASSISTANT PROFESSOR, UNIVERSITY OF BRITISH COLUMBIA


Anchorage
Gulf of Alaska
Hudson Bay
Canada
ALBERTA
MANITOBA
BRITISH COLUMBIA
SASKATCHEWAN
ONTARIO
QUEBEC
WASHINGTON
MONTANA
NORTH DAKOTA
MINNESOTA
Ottawa
Montreal
SOUTH DAKOTA
WISCONSIN
MICHIGAN
Toronto
MAINE
VT
NH
OREGON
IDAHO
WYOMING
NEW YORK
MA
CT RI
Chicago
IOWA
NEBRASKA
ILLINOIS
OHIO
PENN
NEVADA
UTAH
COLORADO
KANSAS
INDIANA
Philadelphia
MD
DE NJ
United States
MISSOURI
WEST VIRGINIA
San Francisco
KENTUCKY
VIRGINIA
CALIFORNIA


Mt Callaghan
Mt Cayley
Wedge Mountain
Whistler
Stein Valley Nlaka'pamux Heritage Park
Lytton
Agate Dot
Lower Nicola
Nicola
Quilchena
Douglas Lake
Merritt
Kanaka Bar
Falls Creek
Keefers
Mehatl Creek Provincial Park
Boothroyd
Garibaldi
Garibaldi Park
Skookumchuck
Mt. Garibaldi
Mt Tantalus
Boston Bar
Kingsvale
Aspen Grove
Brookmere
Thalia
Hells Gate
Chapmans
Spuzzum
Port Douglas
Tipella
Squamish
Woodfibre
Britannia Beach
Allison Lake
Bankeir
Jellicoe
Yale
Mt Judge Howay
Mt Robie Reid
Bear Creek
Choate
Tulameen
Coalmont
Jura
Princeton
Allenby
Copper Mountain
Hedley
Penticton
Summerland
Chute Lake
Peachland
Kelowna
South Kelowna
McCulloch
Lake Country
Fintry
Foothills
Vernon
Coldstream Ranch
The University of British Columbia...
Gambier Island
Lions Bay
Langdale
Gibsons
Hope
Ruby Creek
Harrison Hot Springs
Harrison Mills
Agassiz
Sunshine Valley
Cascade Recreation Area
Golden Ears Provincial Park
Stave Lake
Vancouver
Coquitlam
Surrey
Richmond
McMillan Island
Deroche
Chilliwack
Mission
South Sumas
Delta
Langley
Langley Twp
Abbotsford
Cultus Lake
White Rock
Skagit Valley Provincial Park
EC Manning Provincial Park
Eastgate
Keremeos
Oliver
Inkaneep
Cathedral Provincial Park and Protected Area
Snowy Protected Area
Osoyoos
Layers
Google

Natural Language Processing 4 Software Engineering

User Feedback Analysis

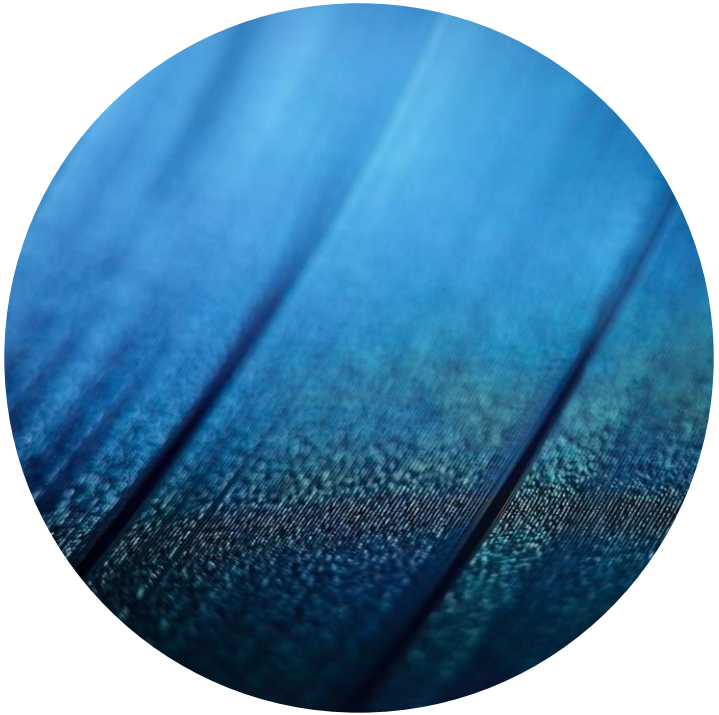
Empirical Studies, MSR

Source Code Rep. Learning

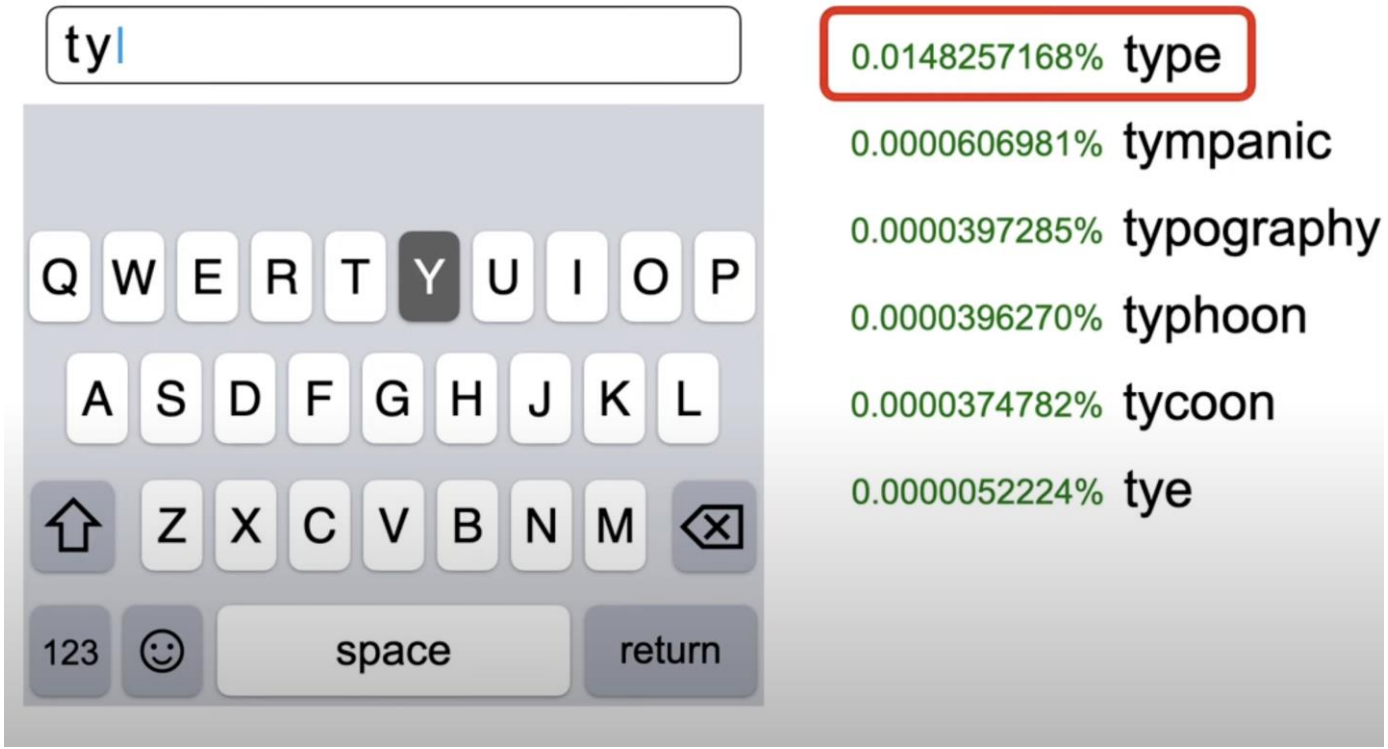
- Comment/Code generation
- Code clone detection

Transfer Learning

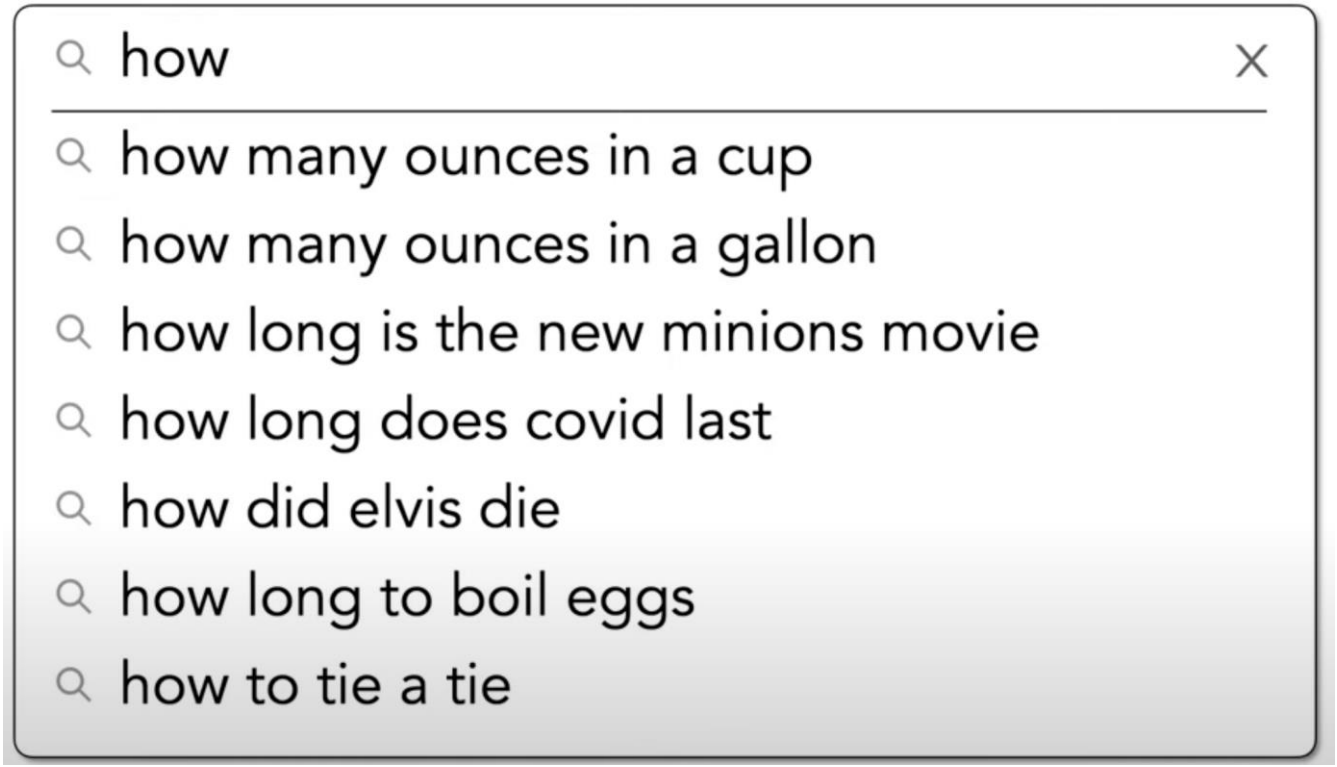
- Transferability of the programming languages
- Few shot learning
- Knowledge transfer among tasks/languages using less computational resources



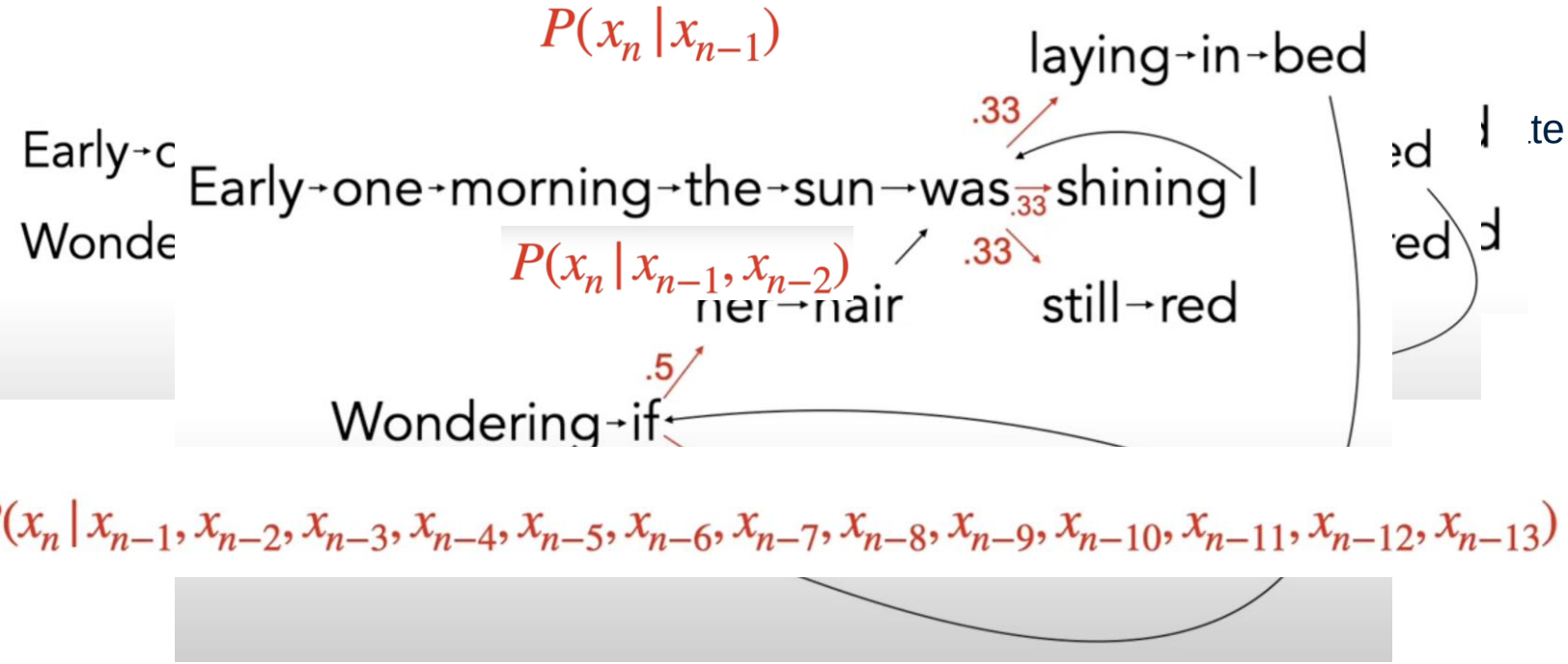
Language Models



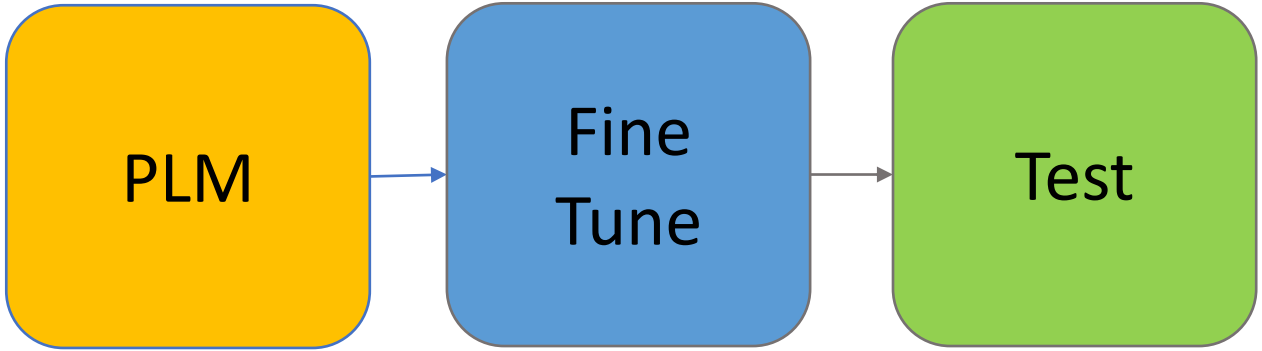
Language Models



Language Models



Finetuning Pre-trained Language Models (PLMs)

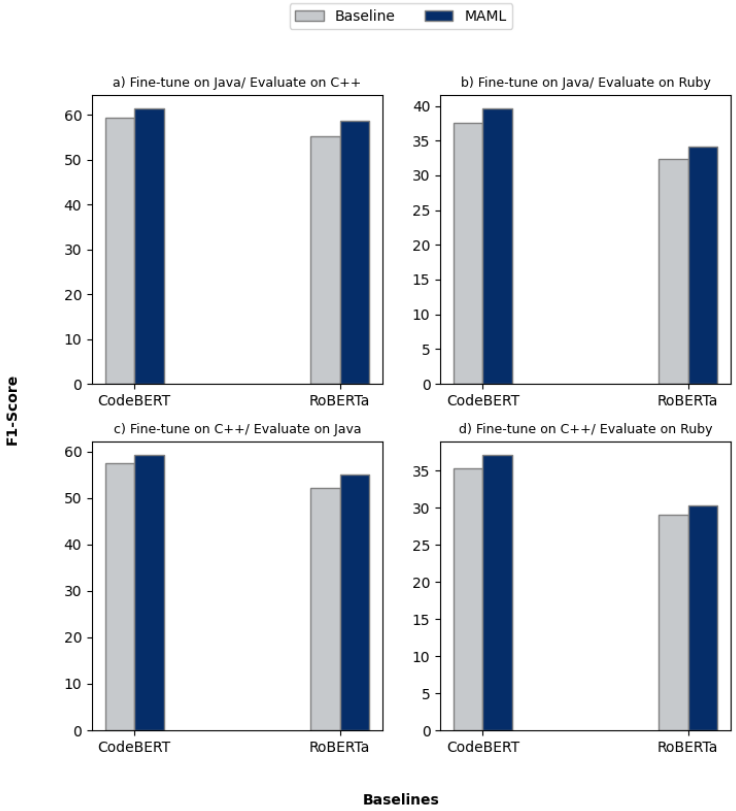


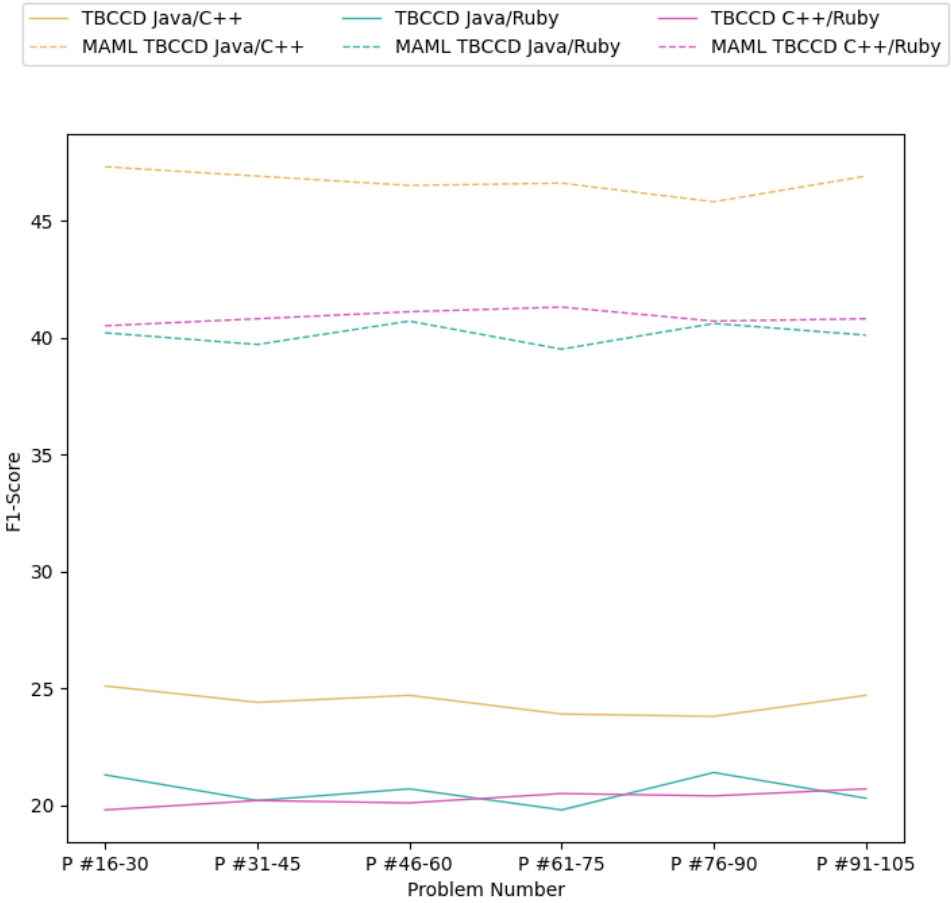
Finetuning PLMs?


```
import collections
import math
import codecs


def _get_ngrams(segment, max_order):
  """Extracts all n-grams upto a given maximum order from an input segment.

  Args:
    segment: text segment from which n-grams will be extracted.
    max_order: maximum length in tokens of the n-grams returned by this
        methods.

  Returns:
    The Counter containing all n-grams upto max_order in segment
    with a count of how many times each n-gram occurred.
  """
  ngram_counts = collections.Counter()
  for order in range(1, max_order + 1):
    for i in range(0, len(segment) - order + 1):
      ngram = tuple(segment[i:i+order])
      ngram_counts[ngram] += 1
  return ngram_counts

```

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 population of developers, code intelligence, which aims to leverage AI to help software developers improve the productivity of the development process, is growing increasingly important in both communities of software engineering and artificial intelligence.

When developers want to find code written by others with the same intent, code search systems can help automatically retrieve semantically relevant code given natural language queries. When developers are confused about what to write next, code completion systems can help by automatically completing the following tokens given the context of the edits being made. When developers want to implement Java code with the same function of some existing body of Python code, code-to-code translation systems can help translate from one programming language (Python) to another (Java).


Baseline
MAML
a) Fine-tune on Java/ Evaluate on C++
b) Fine-tune on Java/ Evaluate on Ruby
c) Fine-tune on C++/ Evaluate on Java
d) Fine-tune on C++/ Evaluate on Ruby
CodeBERT
RoBERTa
F1-Score
Baselines


TBCCD Java/C++
MAML TBCCD Java/C++
TBCCD Java/Ruby
MAML TBCCD Java/Ruby
TBCCD C++/Ruby
MAML TBCCD C++/Ruby
F1-Score
P #16-30
P #31-45
P #46-60
P #61-75
P #76-90
P #91-105
Problem Number

Java

```java
import java.util.Scanner;
public class Main {
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int n =scan.nextInt();
        int a,b,c;
        for(int d = 0;d<n;d++){
            a = scan.nextInt();
            b = scan.nextInt();
            c = scan.nextInt();
            if(a*a+b*b == c*c || b*b+c*c == a*a || c*c+a*a == b*b){
                System.out.println("Yes");
            }else{
                System.out.println("No");
            }
        }
    }
}
```

C++

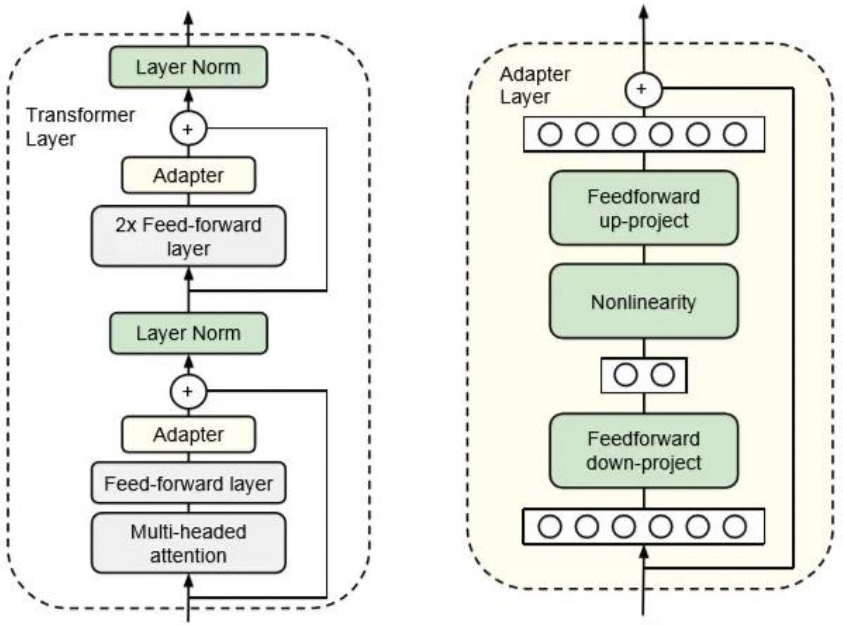
```cpp
#include <iostream>
#include <stdio.h>
#include <string>
#include <math.h>
#include <algorithm>
using namespace std;
int main(){
 int a, b, c, N, i;
 cin >> N;

 for(i = 0;i < N;i++){
  cin >> a >> b >> c;
  if(a*a + b*b == c*c || a*a + c*c == b*b || b*b + c*c == a*a){
   cout << "YES" << endl;
  }
  else cout << "NO" << endl;
 }

 return 0;
}
```

Ruby

```ruby
n = gets.to_i
n.times{
  input = gets.split(" ").map(&:to_i).sort
  if input[2]**2 == input[0]**2 + input[1]**2
    puts "YES"
  else
    puts "NO"
  end
}
```

Adapters

1. **Fine-tuning**
2. **Domain adaptation**



Internal architecture of Transformer blocks using adapters (Left Figure) and internal architecture of language adapters (Right Figure)

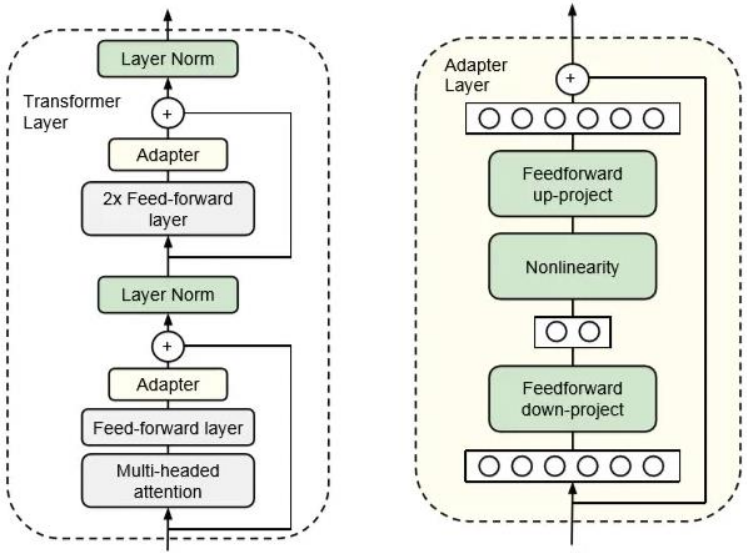
Fig from: N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter efficient transfer learning for nlp," in International Conference on Machine Learning. PMLR, 2019, pp. 2790–2799

Language/Task Adapters

- Language adapters: learn language-specific transformation

- Trained on unlabeled data using an abstract objective function such as mask language modeling (MLM)

- Task adapters: learn about a specific task



Internal architecture of Transformer blocks using adapters (Left Figure) and internal architecture of language adapters (Right Figure)

Fig from: N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly, "Parameter efficient transfer learning for nlp," in International Conference on Machine Learning. PMLR, 2019, pp. 2790–2799

Adapters for SE?

How to?

- Do they work for bimodal transfer?
- How do they perform for code-related tasks?

SE-specific adapters?

- Other purposes of adapters?
- New adapters for source code?

MODE-X

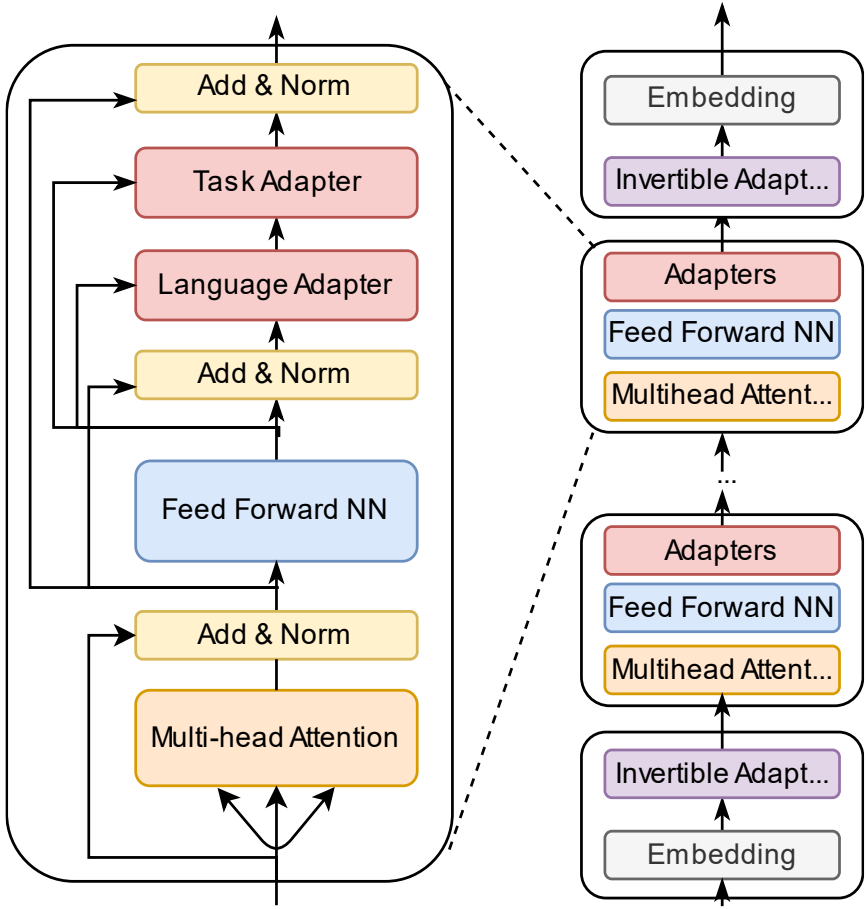
Goel, Divyam, Grover, Ramansh, and Fatemeh H. Fard. On the cross-modal transfer from natural language to code through adapter modules.

MODE-X

- Knowledge transfer from pre-trained models (PLMs) trained on Natural Language to Source Code

- Performance of adapters for code-PLMs

Goel, Divyam, Grover, Ramansh, and Fatemeh H. Fard. On the cross-modal transfer from natural language to code through adapter modules. In Proceedings of the 30th IEEE/ACM International Conference on Program Comprehension, ICPC '22, page 71–81, New York, NY, USA, 2022.

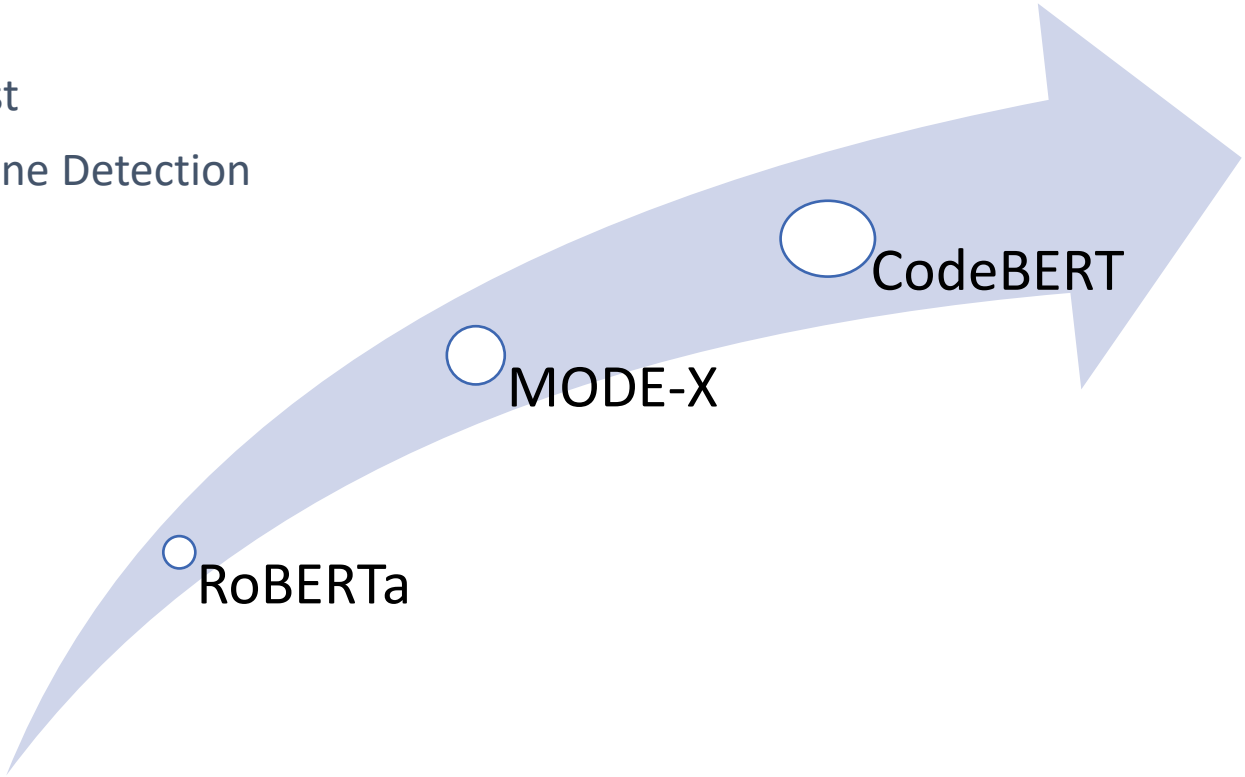


Invertible Adapters

- Handle mismatch between the vocabularies of a PLM and a new unseen language
- Enables efficient utilization of the "parameter budget"

Cloze Test

Code Clone Detection



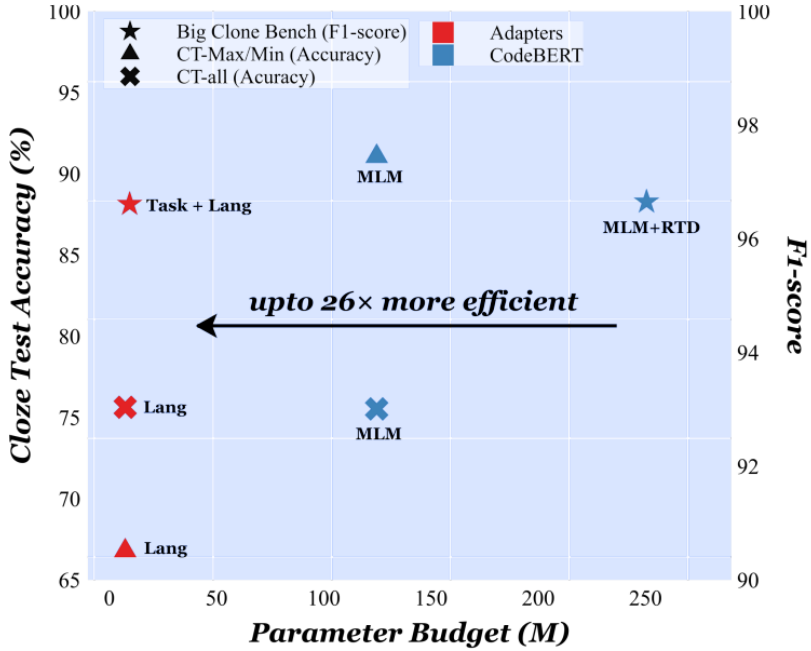
Accuracy Scores on Cloze Test

Model	Python	Java
CT max/min		
RoBERTa	59.18	59.75
RoBERTa+LA	66.30	66.81
CodeBERT	79.27	91.08
CT-all		
RoBERTa	54.49	50.75
RoBERTa+LA	74.35	75.63
CodeBERT	83.33	75.53

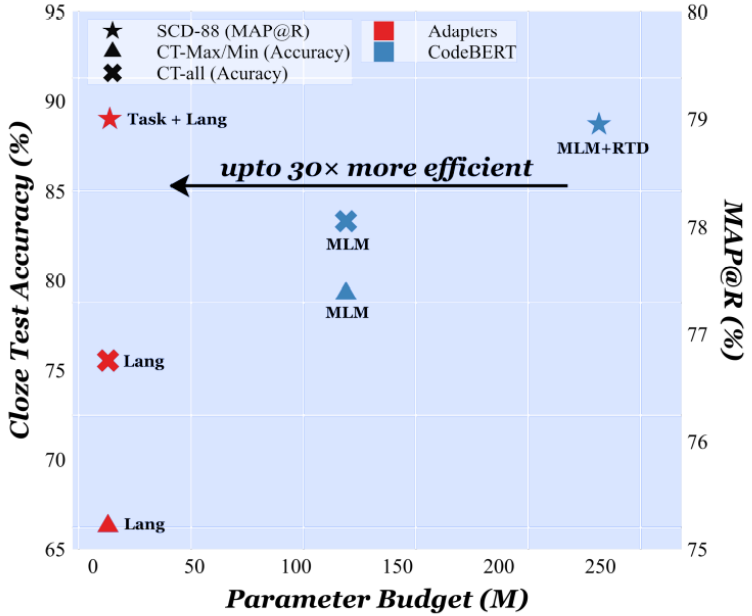
Code Clone Detection Results

Model	Dataset	Score
RoBERTa	POJ-104 (MAP@R)	81.52
MODE-X (C/C++)		82.40
CodeBERT		86.48
RoBERTa	BCB (F1)	95.61
MODE-X (Java)		96.61
CodeBERT		96.65
RoBERTa	SCD-88 (MAP@R)	73.90
MODE-X (Python)		75.65
CodeBERT		78.95

Computational Efficiency of Adapters

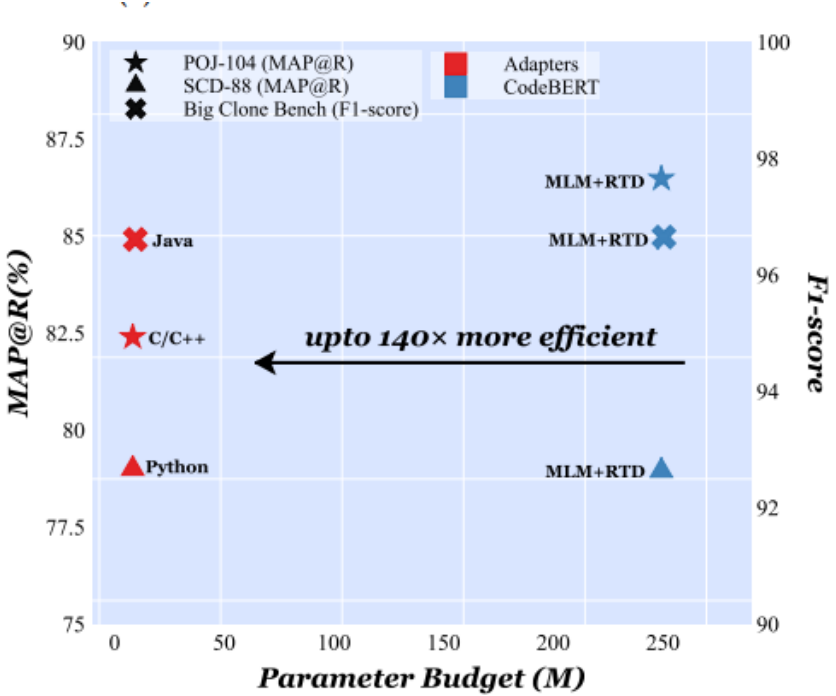


Parameter budget of Java-adapters and CodeBERT in millions



Parameter budget of Python-adapters and CodeBERT in millions

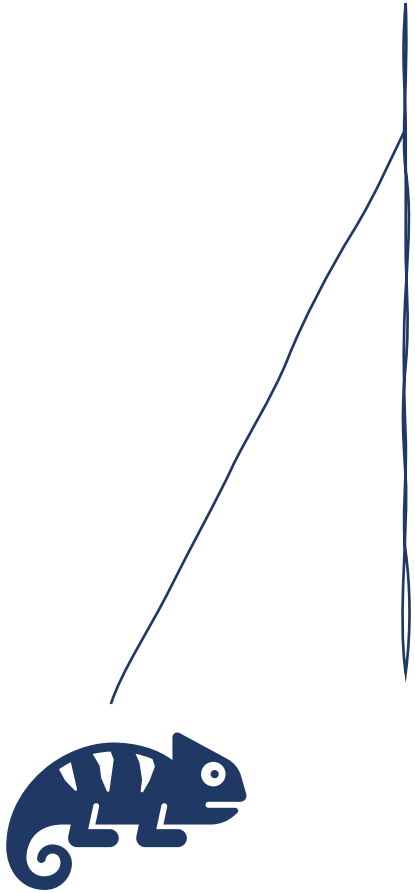
Computational Efficiency of Adapters



Parameter budget of adapters and CodeBERT for code clone detection

Model	Memory (MB)	% Model
CodeBERT	477.98	100
L-adapters	28.20	5.89
T-adapters	3.43	0.72
MODE-X	31.63	6.62

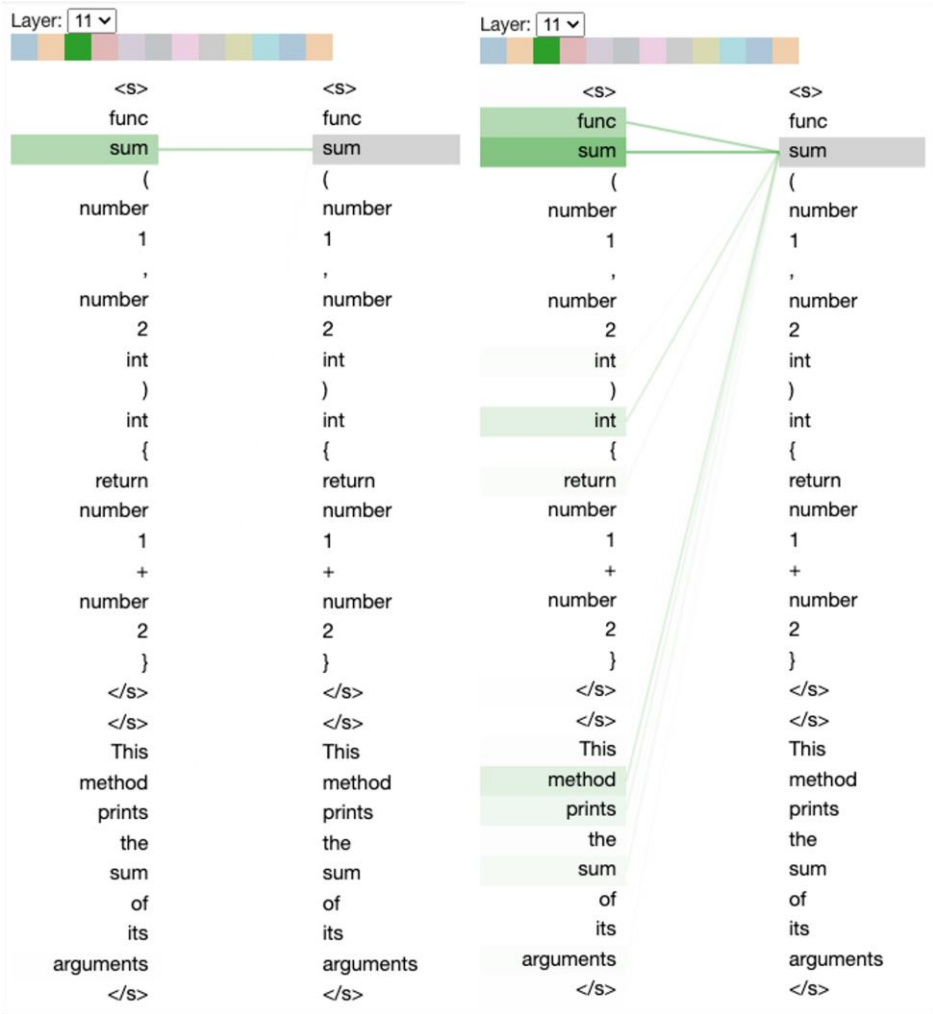
Memory usage of adapters compared to C-PTLM



- Utilize adapters for knowledge transfer from N-PLM to source code (SE-tasks)
- Adapters are more efficient in terms of the number of parameters, memory usage, and inference time.

If we encourage the model weights to be closer to the pre-trained model, we could improve the fine-tuning results without using additional data/parameters.

Go Attention



How to?

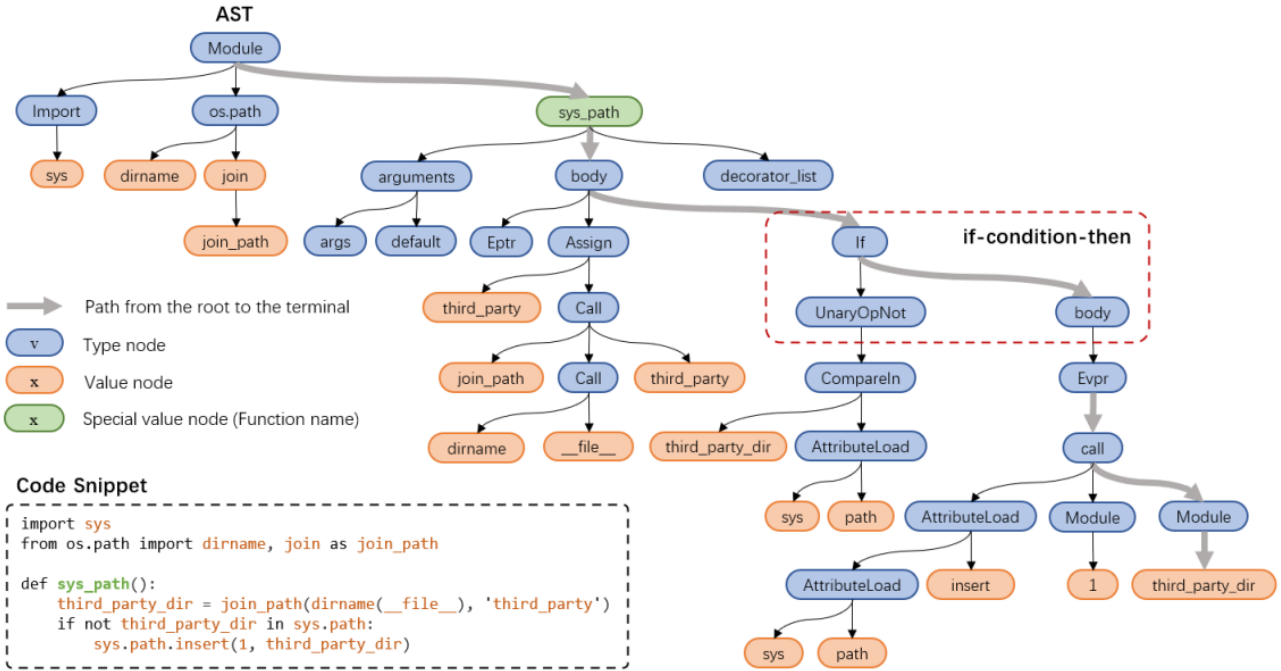
SE-specific adapters?

- Other purposes of adapters?
- New adapters for source Code?

CODEBERTER

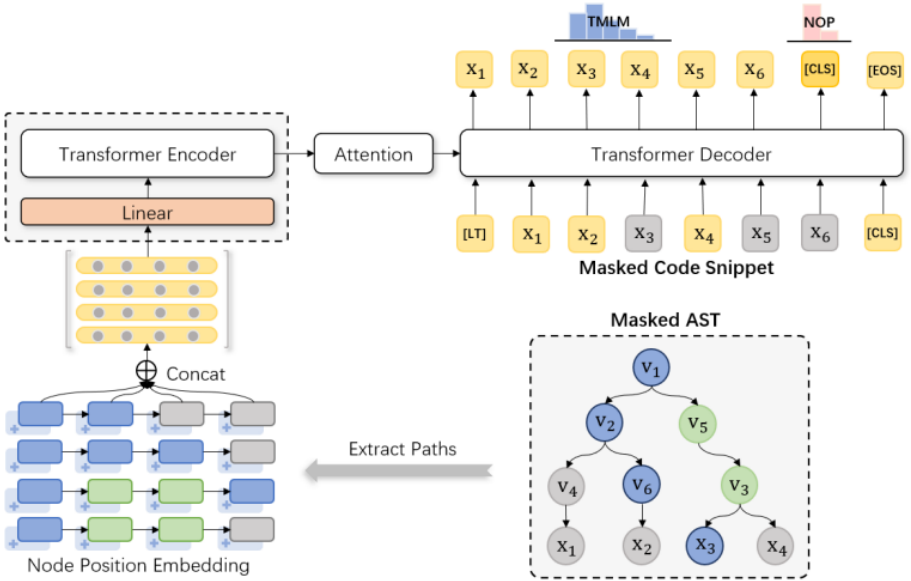
Iman Saberi, Fatemeh H. Fard, Model-Agnostic Syntactical Information for Pre-Trained Programming Language Models.

Code Syntactic Information



Picture from: X. Jiang, Z. Zheng, C. Lyu, L. Li, and L. Lyu, “Treebert: A treebased pre-trained model for programming language,” in Uncertainty in Artificial Intelligence. PMLR, 2021, pp. 54–63.

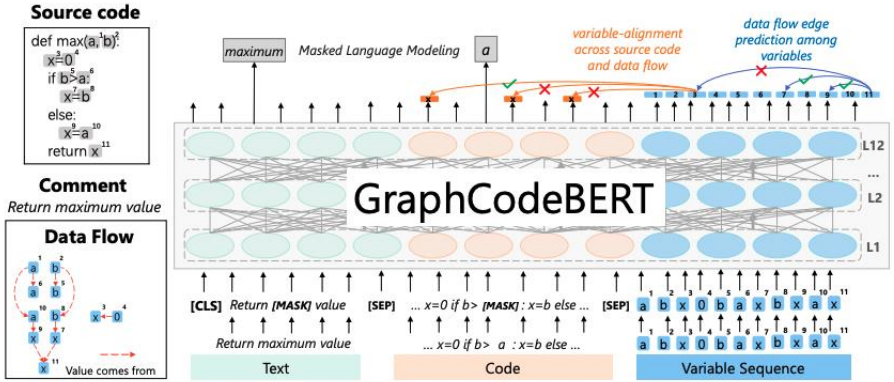
TreeBERT



Scaling input ➡
inefficient computation

Picture from: X. Jiang, Z. Zheng, C. Lyu, L. Li, and L. Lyu, “Treebert: A treebased pre-trained model for programming language,” in Uncertainty in Artificial Intelligence. PMLR, 2021, pp. 54–63.

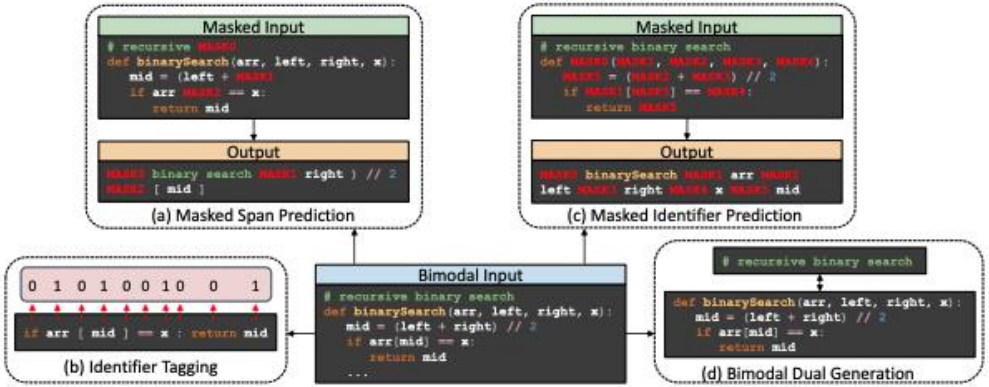
GraphCodeBERT



Picture from: D. Guo, S. Ren, S. Lu, Z. Feng, D. Tang, L. Shujie, L. Zhou, N. Duan, A. Svyatkovskiy, S. Fu et al., "Graphcodebert: Pre-training code representations with data flow," in International Conference on Learning Representations.

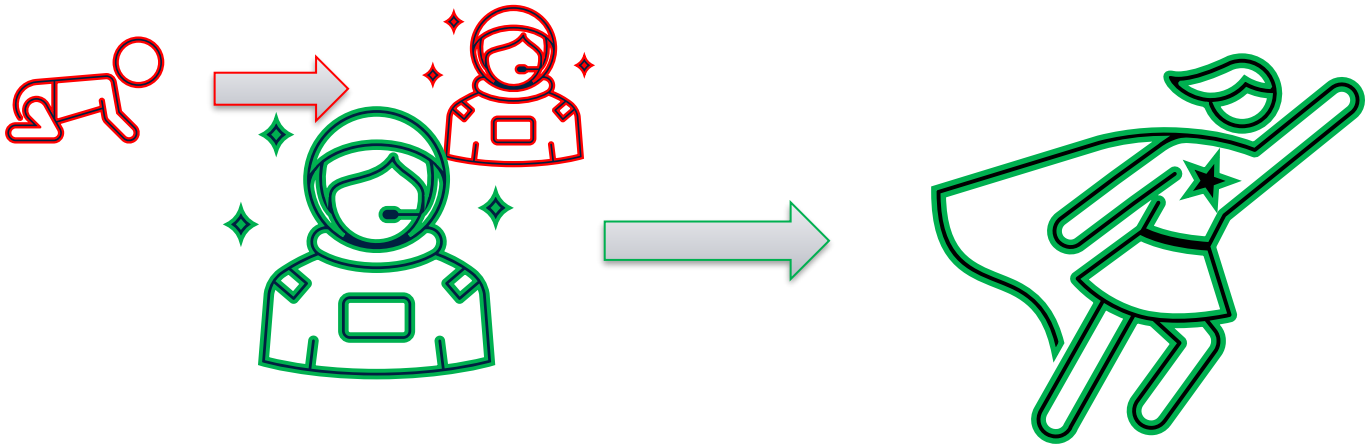
Require Pre-Training

CodeT5



Picture from: Y. Wang, W. Wang, S. Joty, and S. C. Hoi, "Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation," in Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, 2021, pp. 8696–8708.

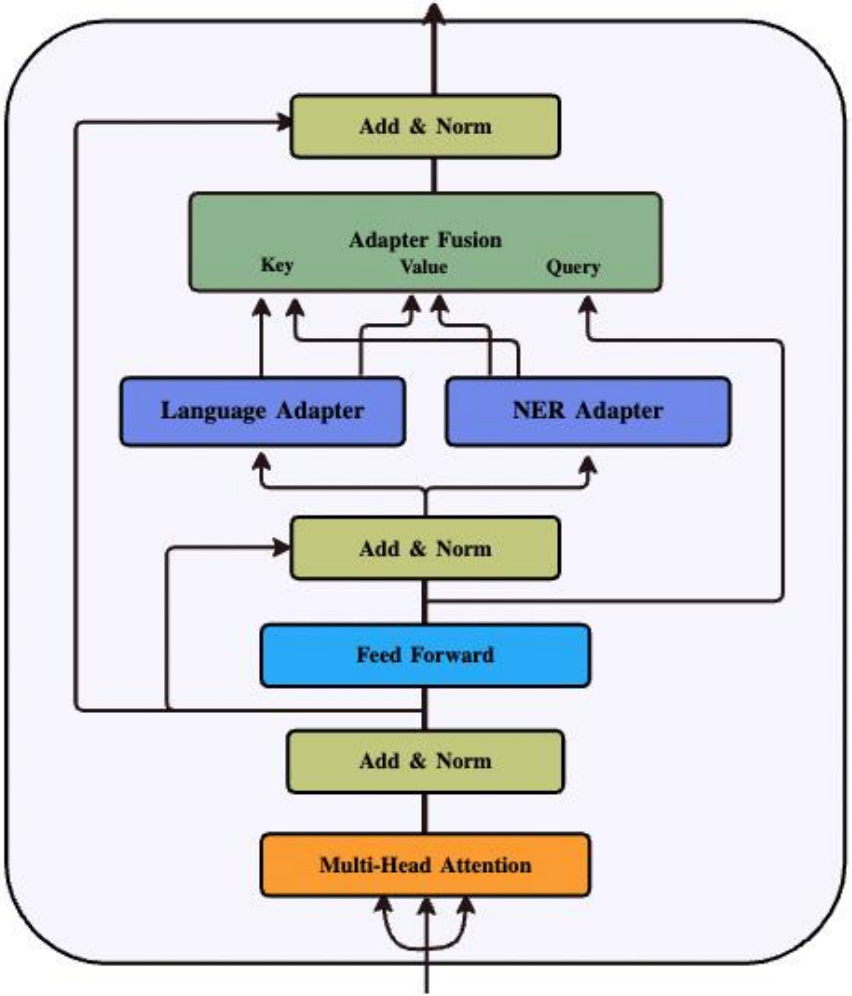
Challenge: How to impose syntactical information of source code to **existing pretrained models**?



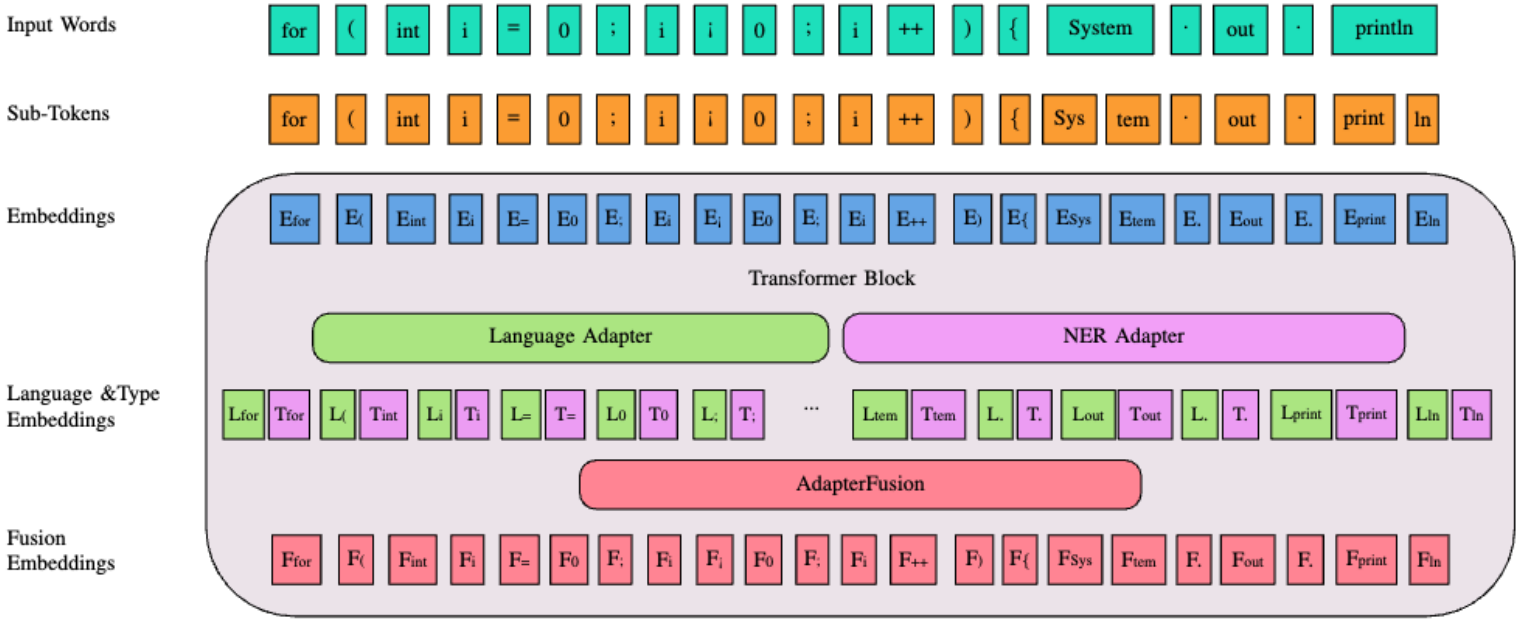
Avoid pre-training while adding new information

NER Adapter

Token Type Classification Loss (TTC)



Overall Architecture



The input data flow for the sample when fed into a transformer block equipped with NER, language and Fusion adapters.

NER Adapter Results

CodeSearchNet data

Language	Precision	Recall	F1-score	Accuracy
Ruby	0.68	0.65	0.66	0.92
JavaScript	0.78	0.79	0.78	0.91
Go	0.78	0.82	0.80	0.94
Python	0.95	0.94	0.94	0.98
Java	0.78	0.79	0.78	0.89

Code Refinement

Identify and fix bugs automatically

Method/Model	BLEUNSEdit	Accuracy
Naïve copy	78.06	0.0
LSTM	76.76	10.0
Transformer	77.21	14.7
RoBERTa (code)	77.30	15.9
CodeBERT	77.42	16.4
CodeBERTER	78.20	17.8
CoText	77.91	22.64
NSEdit	71.06	24.04

Code Summarization

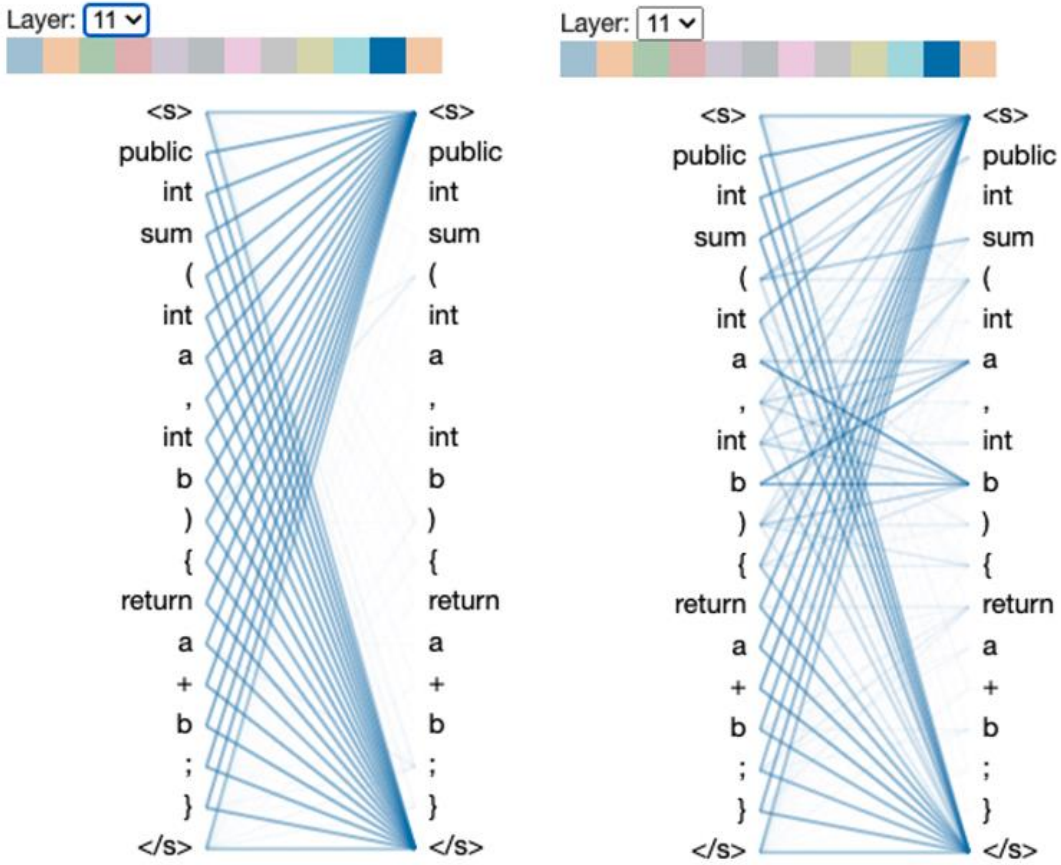
Automatically generating descriptions of the functionality of a given code

Models	Ruby	JavaScript	Go	Python	Java	Average
CodeBERTER	**15.90**	**16.12**	**23.34**	18.38	19.95	**18.738**

Works well for languages with less training data

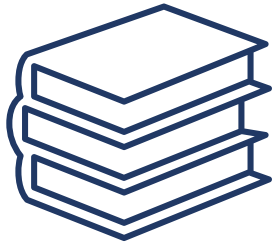
Attention Change with NER Adapter

- **CodeBERTER** (right)
- CodeBERT (figure)



SE-Specific adapters can help:

- Avoid pre-training
- Impose new information
- Improve the results



- Adapters can improve knowledge transfer (bimodal and uni modal)
- Can be used for different purposes other than just fine-tuning and domain-adaptation
- Improve the results (specifically for low resource languages)
- Develop new SE-specific adapters

THE UNIVERSITY OF BRITISH COLUMBIA