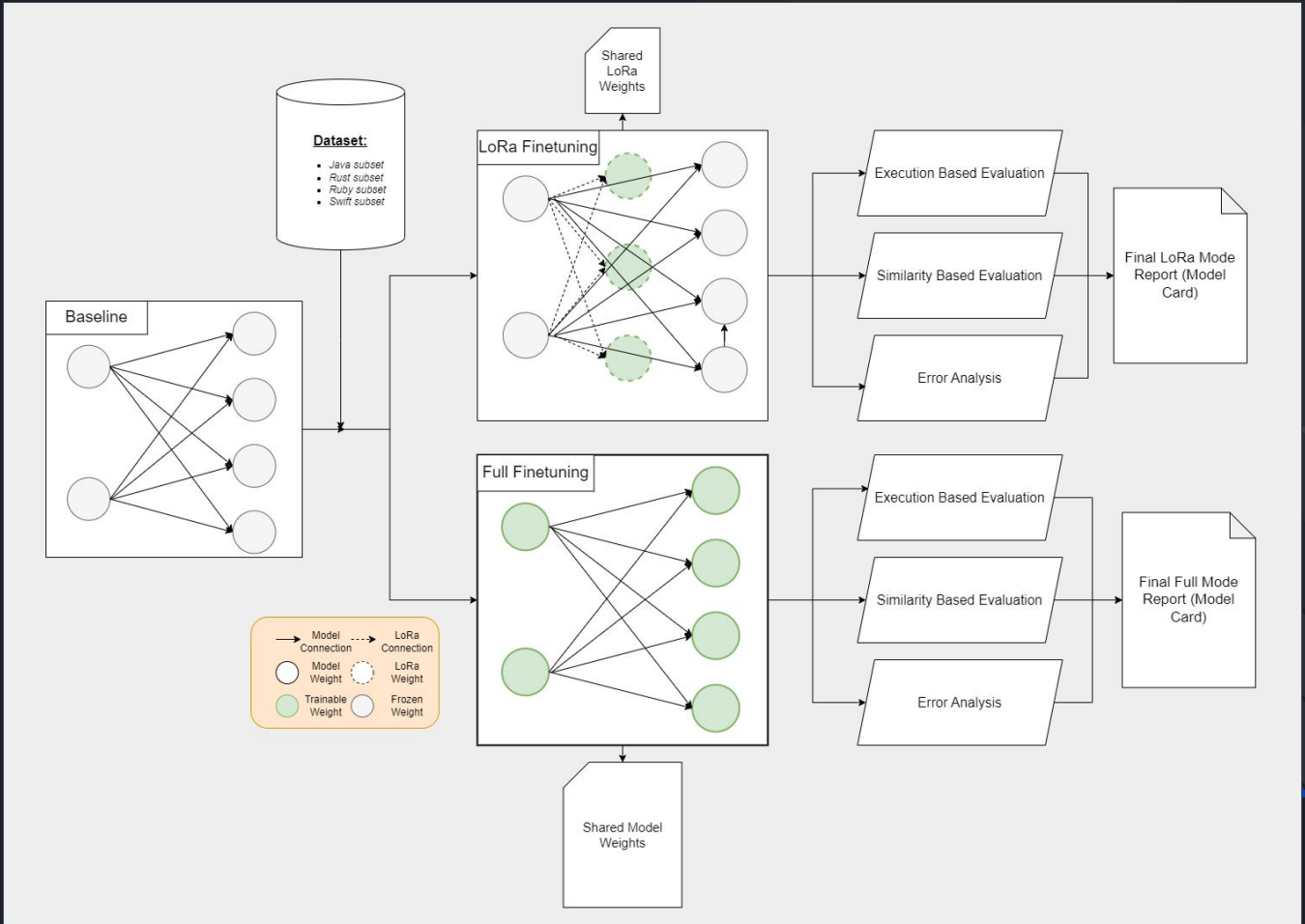
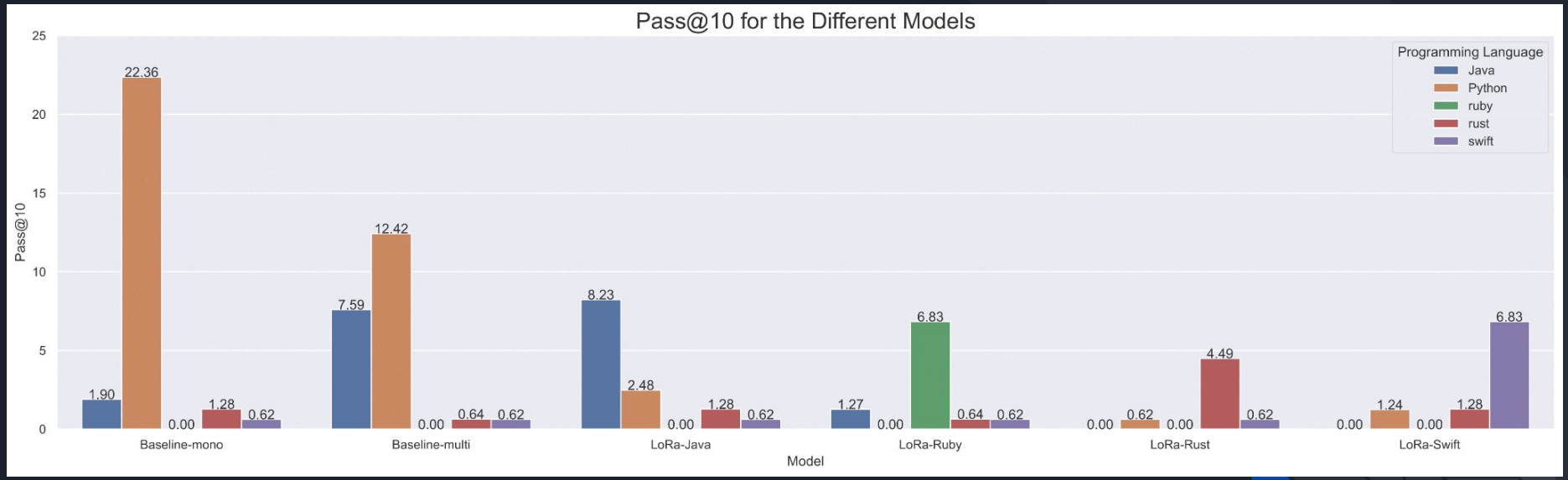
Scaling Multi-Lingual Code Language Models

Towards a more Democratized Access

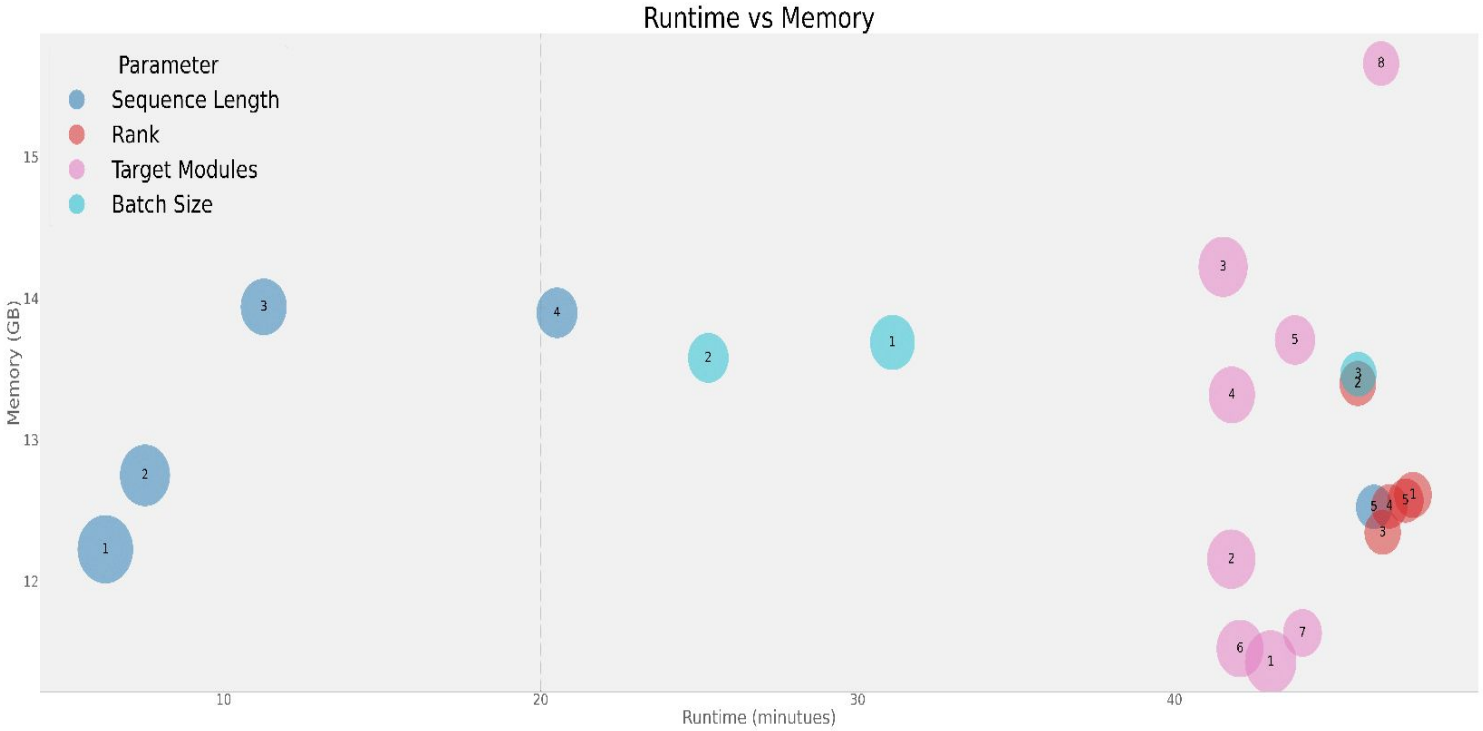
Methodology



Results


Pass@10 for the Different Models
Programming Language
Java
Python
ruby
rust
swift
Pass@10
Model
Baseline-mono
Baseline-multi
LoRa-Java
LoRa-Ruby
LoRa-Rust
LoRa-Swift
1.90
22.36
0.00
1.28
0.62
7.59
12.42
0.00
0.64
0.62
8.23
2.48
0.00
1.28
0.62
1.27
0.00
6.83
0.64
0.62
0.00
0.62
0.00
4.49
0.62
0.00
1.24
0.00
1.28
6.83

Trade-Off Analysis



Error Analysis

```java
import java.util.*;
import java.lang.reflect.*;
import org.javatuples.*;
import java.security.*;
import java.math.*;
import java.io.*;
import java.util.stream.*;
class Problem {
    // Input to this function is a string containing multiple groups of nested parentheses. Your goal is to
    // separate those group into separate strings and return the array list of those.
    // Separate groups are balanced (each open brace is properly closed) and not nested within each other
    // Ignore any spaces in the input string.
    // >>> separateParenGroups(("( ) (( )) (( )( ))"))
    // (new ArrayList<String>(Arrays.asList((String)"()", (String)"(())", (String)"(()())")))
    public static ArrayList<String> separateParenGroups(String paren_string) {
        ArrayList<String> arr = new ArrayList<String>(Arrays.asList(paren_string));
        while (arr.size() > 1) {
            int[] i = new int[arr.size()];
            for (int i_ = 0; i_ < arr.size(); i_++) {
                int[] i_2 = new int[arr.size() - 1];
                for (int i_3 = 0; i_3 < arr.size() - 1; i_3++) {
                    int[] i_4 = new int[arr.size() - 1];
                    for (int i_5 = 0; i_5 < arr.size() - 1; i_5++) {
                        int[] i_6 = new int[arr.size() - 1];
                        for (int i_7 = 0; i_7 < arr.size() - 1; i_7++) {
                            int[] i_8 = new int[arr.size() - 1];
                            for (int i_9 = 0; i_9 < arr.size() - 1; i_9++)
```

Open-Source

Hugging Face | Search models, datasets, users... | Models | Datasets | Spaces | Docs | Solutions | Pricing

ammarnasr / codegen-350M-mono-java | like 0

Text Generation | Adapter Transformers | ammarnasr/the-stack-java-clean | code | License: mit

Model card | Files and versions | Community | Settings | Use in Adapter Transformers

Edit model card

CodeGen (CodeGen-Mono 350M LoRa Java)

Model description

CodeGen LoRa Java is a family of autoregressive language models fine-tuned using LoRa on Different Programming Langauges.

Training data

This model was fine-tuned on the cleaned Java subset from TheStack Avilable here. The data consists of 1 Million Java code files.

Training procedure

This model was fine-tuned using LoRa on 1 T4 GPU. The model was trained for 10,000 steps with batch size of 4. The model was trained using causal language modeling loss.

Evaluation results

We evaluate our models on the MultiPle-E bencchmark. The model achieves 8.9 Pass@10 Rate.

Intended Use and Limitations

However, the model is intended for and best at **program synthesis**, that is, generating executable code given English prompts, where the prompts should be in the form of a comment string. The model can complete partially-generated code in Java and Python.

How to use

Downloads last month
0

Hosted inference API

Text Generation

Inference API does not yet support adapter-transformers models for this pipeline type.

Dataset used to train ammarnasr/codegen-350M-mono-java

ammarnasr/the-stack-java-clean
Viewer • Updated about 13 hours ago • 61