# CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation

Yue Wang[1], Weishi Wang[12], Shafiq Joty[12], Steven Hoi[1]
[1]Salesforce Research Asia [2]Nanyang Technological University

## Overview

### Motivation

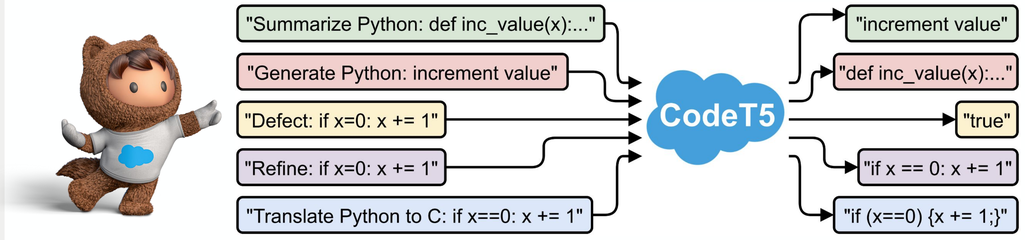
Inspired by the success of pre-trained models in NLP, there is a recent surge of pre-trained programming language models, e.g., CodeBERT and CodeGPT

However, existing programming language models have **two shortcomings**:

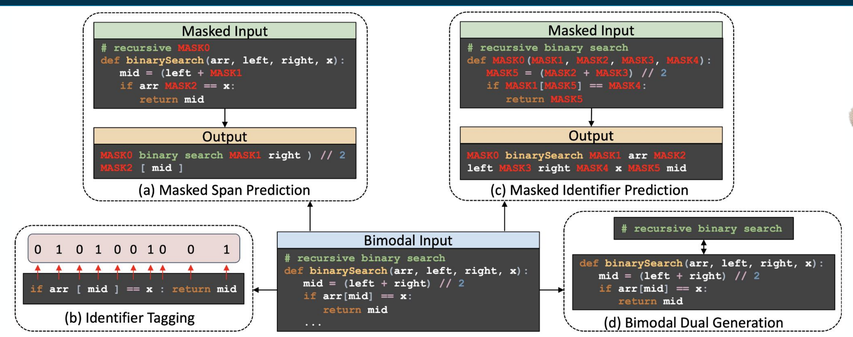
- Most current methods either rely on an encoder-only (or decoder-only) pre-training that is suboptimal for generation (resp. understanding) tasks
- They often process the code as natural language (NL), neglecting special characteristics of programming language (PL) such as token types

### Contributions

- Present **CodeT5**, a novel pre-trained encoder-decoder model that supports both understanding and generation task in multi-task learning
- Propose an *identifier-aware denoising objective* to fuse code token types and a *bimodal dual generation task* to learn a better NL-PL alignment
- CodeT5 yields state-of-the-art results on **14 sub-tasks** in CodeXGLUE




## Pre-training Tasks & Datasets




**Overview of pre-training tasks**: We first alternately train masked span prediction (MSP) , masked identifier prediction (MIP), and identifier tagging (IT) on both unimodal and bimodal data, and then leverage the bimodal data for dual generation (dual-gen) training.

**Pre-training data:** 6 PLs from CodeSearchNet & 2 PLs (C/C#) from Google BigQuery

Data tokenized by **our own pre-trained code-specific tokenizer** (Vocabulary=32K)

- Default T5 tokenizer encodes some common tokens, e.g., '{' and '}', into <unk>
- Reduce the tokenized code length (**30%~45%**) ⇒ accelerate training!

| | PLs | W/ NL | W/o NL | Identifier |
|---|---|---|---|---|
| CodeSearchNet | Ruby | 49,009 | 110,551 | 32.08% |
| | JavaScript | 125,166 | 1,717,933 | 19.82% |
| | Go | 319,132 | 379,103 | 19.32% |
| | Python | 453,772 | 657,030 | 30.02% |
| | Java | 457,381 | 1,070,271 | 25.76% |
| | PHP | 525,357 | 398,058 | 23.44% |
| Our | C | 1M | - | 24.94% |
| | CSharp | 228,496 | 856,375 | 27.85% |
| | Total | 3,158,313 | 5,189,321 | 8,347,634 |

Pre-training data statistics.

## Fine-tuning Results

**Task-specific transfer learning:** fine-tune on a broad set of tasks in CodeXGLUE benchmark

**Generation tasks**
- Summarization (PL→ NL)
- Generation (NL→ PL)
- Refinement (buggy PL → correct PL)
- Translation (PL1 → PL2)

**Multi-task learning:** prepend a unified set of task control prompts, e.g., "Translate Python to C"
- Balanced sampling to avoid the bias towards some tasks
- Allow to select different checkpoints for different tasks

**Understanding tasks**
- Defect detection (PL → 0/1)
- Clone detection (PL1 + PL2 → 0/1)

| Methods | Ruby | JavaScript | Go | Python | Java | PHP | Overall |
|---|---|---|---|---|---|---|---|
| RoBERTa | 11.17 | 11.90 | 17.72 | 18.14 | 16.47 | 24.02 | 16.57 |
| CodeBERT | 12.16 | 14.90 | 18.07 | 19.06 | 17.65 | 25.16 | 17.83 |
| DOBF | - | - | - | 18.24 | 19.05 | - | - |
| PLBART | 14.11 | 15.56 | 18.91 | 19.30 | 18.45 | 23.58 | 18.32 |
| CodeT5-small | 14.87 | 15.32 | 19.25 | 20.04 | 19.92 | 25.46 | 19.14 |
| +dual-gen | 15.30 | 15.61 | 19.74 | 19.94 | 19.78 | 26.48 | 19.48 |
| +multi-task | 15.50 | 15.52 | 19.62 | 20.10 | 19.59 | 25.69 | 19.37 |
| CodeT5-base | 15.24 | 16.16 | 19.56 | 20.01 | 20.31 | 26.03 | 19.55 |
| +dual-gen | **15.73** | 16.00 | 19.71 | 20.11 | 20.41 | **26.53** | 19.75 |
| +multi-task | 15.69 | **16.24** | **19.76** | **20.36** | **20.46** | 26.09 | **19.77** |

Code summarization results (smoothed BLEU-4), "Overall" is the average score over 6 PLs..

| Methods | EM | BLEU | CodeBLEU |
|---|---|---|---|
| GPT-2 | 17.35 | 25.37 | 29.69 |
| CodeGPT-2 | 18.25 | 28.69 | 32.71 |
| CodeGPT-adapted | 20.10 | 32.79 | 35.98 |
| PLBART | 18.75 | 36.69 | 38.52 |
| CodeT5-small | 21.55 | 38.13 | 41.39 |
| +dual-gen | 19.95 | 39.02 | 42.21 |
| +multi-task | 20.15 | 35.89 | 38.83 |
| CodeT5-base | 22.30 | 40.73 | 43.20 |
| +dual-gen | **22.70** | **41.48** | **44.10** |
| +multi-task | 21.15 | 37.54 | 40.01 |

Text-to-code generation results on concode dataset.

**☆Main Observations:**
- CodeT5 significantly outperforms existing baselines in most tasks
- Dual-gen benefits on NL-PL tasks while multi-task benefits on code summarization and refinement

| Methods | Java to C# | | C# to Java | | Refine Small | | Refine Medium | |
|---|---|---|---|---|---|---|---|---|
| | BLEU | EM | BLEU | EM | BLEU | EM | BLEU | EM |
| Naive Copy | 18.54 | 0 | 18.69 | 0 | 78.06 | 0 | 90.91 | 0 |
| RoBERTa (code) | 77.46 | 56.10 | 71.99 | 57.90 | 77.30 | 15.90 | 90.07 | 4.10 |
| CodeBERT | 79.92 | 59.00 | 72.14 | 58.80 | 77.42 | 16.40 | 91.07 | 5.20 |
| GraphCodeBERT | 80.58 | 59.40 | 72.64 | 58.80 | **80.02** | 17.30 | **91.31** | 9.10 |
| PLBART | 83.02 | 64.60 | 78.35 | 65.00 | 77.02 | 19.21 | 88.50 | 8.98 |
| CodeT5-small | 82.98 | 64.10 | 79.10 | 65.60 | 76.23 | 19.06 | 89.20 | 10.92 |
| +dual-gen | 82.24 | 63.20 | 78.10 | 63.40 | 77.03 | 17.50 | 88.99 | 10.28 |
| +multi-task | 83.49 | 64.30 | 78.56 | 65.40 | 77.03 | 20.94 | 87.51 | 11.11 |
| CodeT5-base | **84.03** | **65.90** | **79.87** | **66.90** | 77.43 | 21.61 | 87.64 | 13.96 |
| +dual-gen | 81.84 | 62.00 | 77.83 | 63.20 | 77.66 | 19.43 | 90.43 | 11.69 |
| +multi-task | 82.31 | 63.40 | 78.01 | 64.00 | 78.06 | **22.59** | 88.90 | **14.18** |

Code-to-code translation (Java-C#) and code refinement (small/medium) results.

| Methods | Defect Accuracy | Clone F1 |
|---|---|---|
| RoBERTa | 61.05 | 94.9 |
| CodeBERT | 62.08 | 96.5 |
| DOBF | - | 96.5 |
| GraphCodeBERT | - | 97.1 |
| PLBART | 63.18 | **97.2** |
| CodeT5-small | 63.40 | 97.1 |
| +dual-gen | 63.47 | 97.0 |
| +multi-task | 63.58 | - |
| CodeT5-base | **65.78** | **97.2** |
| +dual-gen | 62.88 | 97.0 |
| +multi-task | 65.02 | - |

Code defect detection & clone detection results.

| Methods | Sum-PY (BLEU) | Code-Gen (CodeBLEU) | Refine Small (EM) | Defect (Acc) |
|---|---|---|---|---|
| CodeT5 | 20.04 | 41.39 | 19.06 | 63.40 |
| -MSP | 18.93 | 37.44 | 15.92 | 64.02 |
| -IT | 19.73 | 39.21 | 18.65 | 63.29 |
| -MIP | 19.81 | 38.25 | 18.32 | 62.92 |

Ablation study on 4 tasks: all objectives contribute to the better performance.

| Type | Code |
|---|---|
| Source | **Text:** returns the string value of the specified field. the value is obtained from whichever scan contains the field.<br>**Env:** Scan s1; Scan s2; boolean hasField |
| CodeT5 | String function (String arg0){<br>if (s1.hasField(arg0))<br>return s1.getString(arg0);<br>else return s2.getString(arg0);} |
| W/o MIP+IT | String function (String arg0){<br>return s1.getString(arg0);} |

Case study on code generation by ablating identifier-aware objectives.

## Conclusion

### Impacts

- Present CodeT5, the code-aware pre-trained encoder-decoder model that yields state-of-the-art results on fourteen sub-tasks in the CodeXGLUE benchmark
- A large pre-trained programming language model with great potential to support a wide range of code intelligence applications in software development lifecycle
- Code and models have been released at github.com/salesforce/CodeT5

### Ethical considerations

**Dataset Bias** **Automation Bias** **Computation** **Security**

Code Blog